Linux
常用命令手册

曹江华 王涛 编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

VISIT…

LANZAROTE Caliente.COM

内 容 简 介

经过十几年的发展，Linux 操作系统不断完善，得到了更多的应用。现在，Linux 的图形化操作界面已经相对成熟，但是命令行技术仍然是 Linux 的核心技术。

本书汇集了 Linux 命令行下核心管理命令（包括最新的 docker 管理命令、systemd 服务管理命令）的功能说明、语法说明、选项介绍、典型示例和注意事项等，对每一个命令都做了非常详尽的介绍，并列举了大量的示例进行说明，可以使读者对 Linux 下的命令有快速深入的认识。全书按照 Linux 命令的功能进行分类，便于读者查询。阅读本书之前不需要读者掌握太多的背景知识，无论读者是 UNIX 用户还是 Linux 新手，甚至是从未接触过 Linux 的 Windows 用户，都可以轻松地理解和掌握这些内容，并可以快速了解和使用各个已发行版本的 Linux 系统。

本书使用 RHEL 7.0 所涉及的命令，同时也适用于其他 Linux 发行版本，是所有 Linux 用户必备的参考用书。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

Linux 常用命令手册 / 曹江华，王涛编著. —北京：电子工业出版社，2015.5

ISBN 978-7-121-25820-6

Ⅰ. ①L… Ⅱ. ①曹… ②王… Ⅲ. ①Linux 操作系统－程序设计－手册 Ⅳ. ①TP316.89-62

中国版本图书馆 CIP 数据核字（2015）第 068772 号

策划编辑：李　冰
责任编辑：徐津平
特约编辑：赵树刚
印　　刷：
装　　订：
出版发行：电子工业出版社
　　　　　北京市海淀区万寿路 173 信箱　　　邮编：100036
开　　本：787×1092　1/16　　　印张：38.5　　　字数：985.6 千字
版　　次：2015 年 5 月第 1 版
印　　次：2015 年 5 月第 1 次印刷
定　　价：95.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件到 dbqq@phei.com.cn。

服务热线：（010）88258888。

前言

Foreword

目前 Linux 操作系统的图形化操作界面已经相当优秀。在 Linux 上可采用多种图形管理程序来改变不同的桌面图案或功能菜单，如 GNOME 和 KDE 等，这一点是 Windows 操作系统单一界面所望尘莫及的。但 Linux 是一个基于命令行的操作系统，命令行的命令是 Linux 操作系统的灵魂和精华所在，只有学会并掌握命令行技术，才能真正精通 Linux，并成为真正的 Linux 高手。命令行能够让用户访问数百个工具软件。在命令行上面，用户可以使用一个管道把工具软件结合在一起执行一项单个工具软件无法完成的工作。

关注命令行

对于某些用户和执行某些任务来说，使用图形界面更容易、更简洁。这实际上依赖于你要做什么和你是谁。图形用户界面和命令行之间的区别就像极品飞车游戏的自动驾驶与自己使用技巧玩一样。不过还是建议你自己慢慢地打任务，这样更有意思，而且你会慢慢地发现自己的技术提高了很多，如果直接下载补丁的话就没有太多的乐趣了。初级 Linux 用户多将矛头指向相对图形界面缺乏的 UNIX 的时候，多少人发出疑问：为什么要死守命令行？我们说一个简单的逻辑：图形界面人机交互好，图形界面简单，图形界面在某些任务方面确实是高效而且简单的，但是，在某些任务方面图形界面是无能的。Linux 操作系统是一款文本式驱动的操作系统，这是它的优点所在，即使是 Windows 处理复杂任务的时候，图形界面也只是一个界面。图形界面消耗的是物理内存的空间，当你的屏幕上需要处理复杂的问题的时候，图形界面要密密麻麻地列出一堆列表，而它的作用不过是基于某个命令的图形外壳，这个时候，图形界面的直观作用和命令行没有区别。而命令行遵循的是逻辑原则，当你需要处理某个任务的时候，命令和参数的位置决定了你执行的逻辑，这个时候图形界面无法做到。也许有人说，对于图形界面的执行逻辑可以决定命令行的位置依赖性，并且可以在合适的地方增加更多、更直观的元素。但是，问题是图形界面一旦达到这个地步，它的定制性能迅速下降。图形界面的优点就在于这里：对于简单、固定逻辑流程的东西，它处理起来简单、直观、高效，所以非常适合作为应用软件的交互界面；但是这也是图形界面的缺点：对于复杂变化逻辑的东西，它的处理能力低下。

本书结构

本书的每一章都建立在前面章节所介绍的知识的基础之上，这样就形成了从始至终的一个自然的学习路径。读者从本书获得了足够的知识之后，也会获得相应的信心，同时会发现章节的顺序和结构就是一个很有用的参考工具。

章　　名	内 容 介 绍
第 1 章　Linux 命令行简介	本章介绍 Linux 的基础知识，包括桌面概述，各种可用的 GUI 应用程序，如何管理用户、目录和文件，以及如何检查系统进程以管理系统资源
第 2 章　Linux 文件管理命令	本章介绍 Linux 层次式文件管理命令
第 3 章　Linux 磁盘管理命令	本章介绍 Linux 磁盘管理和分区及其相关命令
第 4 章　Linux 进程管理命令	本章介绍 Linux 进程管理命令
第 5 章　Linux 网络管理命令	本章介绍 Linux 网络管理命令
第 6 章　Linux 用户管理命令	本章介绍 Linux 用户管理命令
第 7 章　Linux 的备份和压缩命令	本章介绍 Linux 压缩、备份常用命令
第 8 章　Linux 系统管理命令	本章介绍最常用的 Linux 系统管理命令。本章是本书的重点章节
第 9 章　Linux 服务器管理命令	本章介绍 Linux 主流服务器管理命令的使用技巧。本章是本书的重点章节，包括最新的虚拟化工具 docker 管理命令、SELinux 管理命令
第 10 章　Linux 打印管理命令	本章介绍 Linux 打印管理命令的使用方法
第 11 章　Linux 库应用命令	本章介绍 Linux 库的操作命令
第 12 章　Linux 开发应用命令	本章介绍 Linux 的编程环境命令
第 13 章　Linux 行编辑器	本章介绍强大的 Linux 命令行文本编辑器
第 14 章　Linux Shell 命令	一旦读者掌握了 Linux 基础，就可以使用已有的知识，采用 Shell 作为编程语言来构建更加复杂和专用的程序。本章针对 Shell 基础进行学习

本书特点

目前市场上关于 Linux 命令行类的图书并不多，并且大部分只是对命令行的常用命令进行简要介绍，既不全面也不深入，无法满足读者的需求。针对这种现状，本书对 Linux 命令行下的主要命令进行了详尽的系统介绍。

– 命令详尽

本书详细讲解了 492 个 Linux 命令，覆盖了 Linux 应用中所有的核心应用命令，包括最新的 docker 管理命令、systemd 服务管理命令。

– 支持各种环境

本书的编写基于 RHEL7.0 操作系统，汇集了 Linux 命令行下最常用、最重要的命令。这些命令同时也适用于其他 Linux 发行版本，是所有 Linux 用户必备的参考用书。

– 实例丰富

本书中的命令配有详细的实例说明。全书共有 1060 个实例，读者可以根据实例说明进行操作，加深对指令的理解。

– 索引方便

本书提供的命令按照两种方式进行索引：一是按照命令的首字母顺序进行索引；二是按照命令的功能进行索引，方便用户查找使用。

本书适合的读者对象

Linux 应用爱好者，Linux 网络管理员和系统管理员及对故障排除方面感兴趣的读者，IT 行业的相关人员，大专院校计算机专业师生，Linux 系统管理、网络管理、维护的从业人员。作为一本系统管理与维护、网络配置与管理的工具类用书，本书对于相关工程技术人员也是一本不可多得的参考书。

致谢

首先感谢在本书编写过程中领导、朋友和家人的支持及帮助，电子工业出版社的李冰、赵树刚编辑也给予了我无私的帮助和鞭策。本书第 l 章由王涛执笔，第 2 章由张志军、何清、林捷执笔，第 3 章由王波、曹元其、吴少萍、杨水珍执笔，第 4～14 章由曹江华执笔，全书由曹江华进行统稿。另外，杨晓勇帮助完成了资料收集和文字校对工作。由于作者水平有限，书中不足及错误之处在所难免，敬请专家和读者给予批评指正。

编　者

2015 年 4 月于北京

目录

Contents

第 1 章　Linux 命令行简介 1

1.1　Linux 命令行概述 1

1.1.1　Linux 命令行的重要作用 1

1.1.2　Linux 命令行介绍 1

1.1.3　Linux 命令行的启动及退出 2

1.1.4　Linux 命令行提示符 2

1.1.5　命令行的历史记录和编辑 3

1.1.6　命令行快捷键 4

1.2　在命令行下使用 Linux 帮助信息 4

1.2.1　使用 help 命令获得 bash 的内部命令帮助 5

1.2.2　使用 man 获取帮助信息 6

1.2.3　使用 info 获取帮助信息 8

1.2.4　用发行版的系统手册 10

1.2.5　从 Internet 获得帮助 10

第 2 章　Linux 文件管理命令 12

2.1　ls：显示文件名 12

2.2　cat：显示文本文件内容 16

2.3　rm：删除文件 17

2.4　less：分屏显示文件 18

2.5　cp：复制文件 21

2.6　mv：更改文件名 22

2.7　grep：查找字符串 23

2.8　head：显示文件头部 26

2.9　tail：显示文件尾部 27

2.10　sort：按顺序显示文件内容 28

2.11　uniq：忽略文件中的重复行 30

2.12　diff：比较两个文件 32

2.13 diffstat：diff 结果的统计信息 36
2.14 file：测试文件内容 36
2.15 echo：显示文本 37
2.16 date：显示日期和时间 38
2.17 script：记录 Linux 会话信息 40
2.18 apropos：搜索关键字 42
2.19 locate：搜索文件 42
2.20 rmdir：删除目录 44
2.21 chattr：改变文件的属性 44
2.22 cksum：文件的 CRC 校验 45
2.23 cmp：比较文件差异 47
2.24 split：分割文件 47
2.25 dirname：显示文件除名字外的路径 48
2.26 find：查找目录或者文件 48
2.27 findfs：通过列表或用户 ID 查找文件系统 50
2.28 ln：链接文件或目录 51
2.29 lndir：链接目录内容 53
2.30 lsattr：显示文件属性 53
2.31 od：输出文件内容 54
2.32 paste：合并文件的列 55
2.33 stat：显示 inode 内容 56
2.34 tee：读取标准输入到标准输出并可保存为文件 58
2.35 tmpwatch：删除临时文件 59
2.36 touch：更新文件目录时间 59
2.37 tree：以树状图显示目录内容 60
2.38 umask：指定在建立文件时预设的权限掩码 62
2.39 chmod：设置文件或目录的访问权限 63
2.40 chgrp：改变文件或目录所属的群组 66
2.41 chown：改变文件的拥有者或群组 66
2.42 more：查看文件的内容 67
2.43 md5sum：MD5 函数值计算和检查 69
2.44 awk：模式匹配语言 70
2.45 wc：输出文件中的行数、单词数、字节数 75
2.46 comm：比较排序文件 75
2.47 join：将两个文件中指定栏位内容相同的行连接起来 76
2.48 fmt：编排文本文件 77

2.49 tr：转换字符 78
2.50 col：过滤控制字符 80
2.51 colrm：删除指定的行 80
2.52 fold：限制文件列宽 80
2.53 iconv：转换给定文件的编码 81
2.54 dc：任意精度的计算器 82
2.55 expr：求表达式变量的值 82
2.56 strings：显示文件中的可打印字符 84
2.57 xargs：从标准输入读入参数 84
2.58 sum：计算文件的校验和，以及文件占用的块数 86
2.59 setfacl：设定文件访问控制列表 87
2.60 getfacl：获取文件访问控制列表 88
2.61 chacl：更改文件或目录的访问控制列表 89

第 3 章 Linux 磁盘管理命令 91

3.1 df：显示报告文件系统磁盘使用信息 91
3.2 du：显示目录或者文件所占的磁盘空间 92
3.3 dd：磁盘操作 94
3.4 fdisk：磁盘分区 96
3.5 mount：挂载文件系统 99
3.6 umount：卸载文件系统 102
3.7 mkfs：建立各种文件系统 102
3.8 mkfs.ext2：建立一个 Ext2/Ext3 文件系统 104
3.9 mkbootdisk：建立启动盘 105
3.10 fsck：检查文件系统 105
3.11 blockdev：从命令行调用区块设备控制程序 106
3.12 hdparm：设置磁盘参数 107
3.13 mkswap：建立交换分区 109
3.14 swapon：使用交换空间 110
3.15 swapoff：关闭系统交换分区 111
3.16 sync：写入磁盘 111
3.17 e2label：设置卷标 112
3.18 badblocks：检查磁盘 112
3.19 quota：显示磁盘已使用的空间与限制 113
3.20 quotacheck：检查磁盘的使用空间与限制 113
3.21 quotaoff：关闭磁盘空间限制 114

3.22 quotaon：开启磁盘空间限制 114
3.23 quotastats：显示磁盘空间限制 115
3.24 repquota：检查磁盘空间限制的状态 116
3.25 mdadm：RAID 设置工具 116
3.26 tune2fs：文件系统调整 119
3.27 mkisofs：建立 ISO 9660 映象文件 120
3.28 cfdisk：磁盘分区 121
3.29 sfdisk：硬盘分区工具程序 123
3.30 parted：磁盘分区工具 124
3.31 mkinitrd：建立要载入 ramdisk 的映象文件 126
3.32 ssm：命令行集中存储管理工具 126
3.33 使用 xfs 管理命令 129
3.34 LVM 命令列表 134

第 4 章 Linux 进程管理命令 141

4.1 accton：打开或关闭进程统计 141
4.2 lastcomm：显示以前使用过的命令的信息 142
4.3 sa：报告、清理并维护进程统计文件 143
4.4 at：定时运行命令 144
4.5 atq：显示目前使用 at 命令后待执行的命令队列 145
4.6 atrm：删除 at 命令中待执行的命令队列 146
4.7 batch：在系统负载水平允许的时候执行命令 146
4.8 bg：后台运行命令 147
4.9 fg：挂起程序 147
4.10 jobs：显示后台程序 148
4.11 kill：杀掉进程 149
4.12 crontab：设置计时器 150
4.13 ps：查看权限 151
4.14 pstree：显示进程状态树 153
4.15 top：显示进程 154
4.16 nice：改变优先权等级 156
4.17 renice：修改优先权等级 156
4.18 sleep：暂停进程 157
4.19 nohup：用户退出系统之后继续工作 158
4.20 pgrep：查找匹配条件的进程 158
4.21 fuser：用文件或者套接口表示进程 158

4.22 chkconfig：设置系统的各种服务 160
4.23 strace：跟踪一个进程的系统调用或信号产生的情况 162
4.24 ltrace：跟踪进程调用库函数的情况 162
4.25 vmstat：报告虚拟内存统计信息 162
4.26 mpstat：监测 CPU（包括多 CPU）性能 164
4.27 iostat：监测 I/O 性能 165
4.28 sar：系统活动情况报告 166
4.29 pidof：查找正在运行进程的进程 ID（PID） 167
4.30 ntsysv：设置系统服务 168

第 5 章 Linux 网络管理命令 170

5.1 arp：管理系统中的 ARP 高速缓存 170
5.2 arpwatch：监听 ARP 记录 171
5.3 arping：发送 ARP 请求到一个相邻主机 172
5.4 arpd：收集免费 arp 信息的一个守护进程 173
5.5 finger：查找并显示用户信息 173
5.6 ifconfig：设置网络接口 174
5.7 iwconfig：设置无线网卡 176
5.8 iw：新一代无线网络配置工具 180
5.9 hostname：显示主机名 187
5.10 ifup：激活网络设备 188
5.11 ifdown：禁用网络设备 189
5.12 mii-tool：调整网卡模式 189
5.13 route：设置路由表 190
5.14 netstat：查看网络连接 192
5.15 ping：检测主机的连通性 194
5.16 minicom：设置调制解调器 195
5.17 pppd：建立 PPP 连接 196
5.18 pppstats：显示 PPP 连接状态 198
5.19 chat：拨号命令 198
5.20 traceroute：检查数据包所经过的路由 199
5.21 tracepath：追踪连接到目标地址所经过的路由 200
5.22 rcp：远程复制 201
5.23 tcpdump：网络数据分析器 202
5.24 ipcalc：IP 地址计算器 205
5.25 netreport：监视网络状态 206

5.26 ip：网络集成命令工具 206
5.27 pppoe-setup：设置 ADSL 208
5.28 pppoe-start：激活 ADSL 连接 210
5.29 pppoe-stop：断开 ADSL 连接 211
5.30 pppoe-status：检测 ADSL 连接状态 211
5.31 wget：下载文件 212
5.32 ngrep：监控网络接口 215
5.33 lsof：查看打开的文件 217
5.34 ethtool：查询及设置网卡参数 219
5.35 netconf：设置各项网络功能 220
5.36 tc：显示和维护流量控制设置 222
5.37 telnet：远程登录 223
5.38 rlogin：远程登录命令 225
5.39 rsh：远程登录的 Shell 226
5.40 usernetctl：让普通用户控制网络接口 227
5.41 nmcli：NetworkManager 命令行网络接口配置工具 228
5.42 nmtui：基于 Curses 的用户界面 nmtui 232
5.43 nc：netcat，Linux 下用于调试和检查网络的工具包 235
5.44 lnstat：显示 Linux 系统的网络状态 236
5.45 ss：显示网络状态信息 236
5.46 rexec：在指定的远程 Linux 系统主机上执行命令 237

第 6 章 Linux 用户管理命令 239

6.1 useradd：建立用户 239
6.2 userdel：删除用户 241
6.3 usermod：修改已有用户的信息 241
6.4 passwd：设置密码 242
6.5 chage：密码老化 243
6.6 groupadd：添加组 244
6.7 groupdel：删除组账户 244
6.8 groupmod：修改组 245
6.9 vipw：编辑/etc/passwd 文件 245
6.10 vigr：编辑/etc/group 文件 246
6.11 newgrp：转换组 247
6.12 groups：显示组 247
6.13 gpasswd：添加组 248

6.14 whoami：显示当前用户名称 248
6.15 who：显示登录用户 248
6.16 id：显示用户信息 249
6.17 su：切换身份 250
6.18 pwck：检测账户 251
6.19 grpck：检测用户组账号信息的完整性 251
6.20 chsh：设置 Shell 252
6.21 chfn：修改用户信息 253
6.22 ac：显示用户在线时间的统计信息 254
6.23 grpconv：开启群组的投影密码 255
6.24 grpunconv：关闭群组的投影密码 256
6.25 lastlog：显示最近登录用户的用户名、登录端口和 登录时间 256
6.26 logname：显示当前用户的名称 257
6.27 users：显示当前登录到系统的用户 257
6.28 lastb：显示登录系统失败用户的相关信息 258

第 7 章 Linux 的备份和压缩命令 260

7.1 tar：备份文件 260
7.2 dump：备份文件系统 265
7.3 cpio（copy in/out）：建立、还原备份文件 266
7.4 restore：还原备份下来的文件或整个文件系统（一个分区） 268
7.5 bunzip2：解压缩.bz2 文件 269
7.6 bzip2：解压缩.bz2 文件 270
7.7 bzgrep：使用正则表达式搜索.bz2 压缩包中的文件 270
7.8 unzip：解压缩.zip 文件 271
7.9 bzip2recover：修复损坏的.bz2 文件 272
7.10 gzip：压缩文件 272
7.11 compress：压缩、解压文件 273
7.12 gzexe：压缩执行文件 274
7.13 lha：压缩或解压缩文件 275
7.14 unarj：解压缩文件 276
7.15 zip：压缩文件 277
7.16 zipinfo：显示压缩文件的信息 278

第 8 章 Linux 系统管理命令 280

8.1 apmd：高级电源管理 280
8.2 apmsleep：APM 进入休眠状态 281

8.3 apropos：查找使用手册的名字和相关描述 281
8.4 arch：输出主机的体系结构 282
8.5 alias：设置别名 283
8.6 cd：切换目录 283
8.7 clear：清空终端屏幕 284
8.8 clock：系统 RTC 时间设置 284
8.9 cal：显示日历 285
8.10 chroot：改变根目录 286
8.11 date：显示或设置系统时间 287
8.12 dmesg：显示开机信息 288
8.13 dircolors：设置 ls 命令在显示目录或文件时所用的色彩 290
8.14 depmod：分析模块 290
8.15 echo：显示文本行 291
8.16 exec：执行完命令后交出控制权 292
8.17 exit：退出 Shell 292
8.18 eject：弹出介质 292
8.19 enable：启动或关闭 Shell 命令 294
8.20 fc：修改或执行命令 294
8.21 fgconsole：显示虚拟终端的数目 295
8.22 free：显示内存信息 295
8.23 fwhois：显示用户的信息 296
8.24 getty：设置终端模式 296
8.25 gitps：显示程序情况 297
8.26 logwatch：可定制和可插入式的日志监视系统 297
8.27 logsave：把一个命令的输出输出到一个指定的日志 文件中 298
8.28 GRUB：引导加载程序 299
8.29 halt：关闭系统 303
8.30 history：显示历史命令 304
8.31 hwclock：显示与设定硬件时钟 305
8.32 init：进程处理初始化 306
8.33 last：显示登录用户信息 307
8.34 lilo：引导管理器 308
8.35 login：登录系统 309
8.36 local：显示本地支持的语言系统信息 309
8.37 logout：退出系统 311
8.38 logrotate：处理 Log 文件 311

8.39 lsmod：显示 Linux 内核的模块信息 311
8.40 man：格式化和显示在线手册 312
8.41 manpath：设置 man 手册的查询路径 314
8.42 modinfo：显示内核信息 314
8.43 modprobe：自动处理可载入模块 315
8.44 pmap：显示程序的内存信息 316
8.45 procinfo：显示系统状态 317
8.46 pwd：显示工作目录 318
8.47 reboot：重新启动系统 318
8.48 rlogin：远程登录 319
8.49 rmmod：删除模块 319
8.50 rpm：软件包管理 320
8.51 shutdown 命令：系统关机命令 325
8.52 suspend：暂停执行 Shell 325
8.53 nproc：打印当前进程可用的处理器数 326
8.54 tload：显示系统负载 326
8.55 uname：显示系统信息 326
8.56 authconfig：配置系统的认证信息 327
8.57 declare：显示或者设定 Shell 变量 329
8.58 export：设置或者显示环境变量 330
8.59 hostid：打印出当前主机的标识 331
8.60 insmod：载入模块 331
8.61 rdate：显示其他主机的日期与时间 332
8.62 runlevel：显示执行等级 332
8.63 set：设置 Shell 333
8.64 setenv：查询或显示环境变量 334
8.65 setserial：设置或显示串口的相关信息 334
8.66 setup：设置公用程序 335
8.67 symlinks：维护符号链接的工具程序 336
8.68 swatch：系统监控程序 336
8.69 sync：将内存缓冲区内的数据写入磁盘 337
8.70 startx：启动 X Window 337
8.71 sysctl：设置系统核心参数 338
8.72 timeconfig：设置时区 338
8.73 ulimit：控制 Shell 程序的资源 339
8.74 unalias：删除别名 341

8.75　unset：删除变量或函数 341
8.76　up2date：软件包升级 341
8.77　uptime：告知系统运行了多长时间 342
8.78　mouseconfig：设置鼠标相关参数 343
8.79　bind：显示或设置键盘按键及其相关的功能 343
8.80　kbdconfig：设置键盘类型 344
8.81　snapscreenshot：命令行截图 344
8.82　mt：磁带机控制 345
8.83　cdrecord：CD 刻录工具 346
8.84　dvdrecord：DVD 刻录工具 347
8.85　lspci：查看硬件插槽 348
8.86　sane-find-scanner：扫描仪搜索 349
8.87　scanimage：检测扫描仪型号 350
8.88　mtools：命令集 351
8.89　whereis：查找文件 352
8.90　ytalk：与其他用户交谈 352
8.91　apt：软件包在线管理 353
8.92　yum：在线管理软件包 355
8.93　vlock：锁定终端 359
8.94　wait：等待程序返回状态 359
8.95　watch：将结果输出到标准输出设备 360
8.96　rsync：远程数据同步工具 360
8.97　&：将任务放在后台执行 361
8.98　screen：多重视窗管理程序 361
8.99　lsb_release：显示 LSB 和特定版本的相关信息 363
8.100　lscpu：查看 CPU 信息 364
8.101　blkid：查看块设备 365
8.102　journalctl：日志管理 366
8.103　lsblk：列出块设备 369
8.104　systemd：服务管理命令组 370
8.105　GRUB2 378

第 9 章　Linux 服务器管理命令 382

9.1　Apache 服务器应用命令 382
9.1.1　Apache 服务器管理命令 382
9.1.2　Apache 日志管理命令 393

9.2 DNS 服务器管理命令 399
9.2.1 named：域名服务器管理命令 399
9.2.2 rndc：DNS 服务器控制 400
9.2.3 named-checkconf：检查 DNS 配置 401
9.2.4 named-checkzone：检查区域文件的合法性 402
9.2.5 dig：发送域名查询信息包到域名服务器 402
9.2.6 nslookup：交互式查询名称服务器 406
9.2.7 host：使用域名服务器查询主机名字 408
9.2.8 dnssec-keygen：DNSSEC 密钥生成工具 409
9.2.9 dnssec-signkey：DNSSEC 密钥集签名工具 410
9.2.10 dnssec-makekeyset：DNSSEC 区域签名工具 411
9.2.11 dnssec-signzone：DNSSEC 区域签名工具 412
9.2.12 dlint：Bind DNS 服务器辅助工具 413
9.2.13 dnstop：Bind DNS 服务器辅助工具 414
9.3 NFS 服务器管理命令 415
9.3.1 nfsd：启动、停止 NFS 服务器 415
9.3.2 portmap：将 RPC 程序号转换为因特网端口号 415
9.3.3 rpcinfo：报告远程过程调用（RPC）服务器的状态 416
9.3.4 showmount：显示远程已安装文件系统的所有客户机的列表 417
9.3.5 umount：删除当前已挂载的远程文件系统 418
9.3.6 mount：将已命名的文件系统连接到指定的挂载点 419
9.3.7 automount：安装自动安装点 419
9.3.8 mountall：挂载一组文件系统 420
9.3.9 exportfs：重新分享/etc/exports 变更的目录资源 420
9.4 Samba 服务器管理命令 421
9.4.1 启动 Samba 服务器 421
9.4.2 testparm：检查 smb.conf 配置文件的内部正确性 421
9.4.3 smbd(samba daemon)：Samba 服务器程序 422
9.4.4 smbclient：类似 FTP 操作方式的访问 SMB/CIFS 服务器 资源的客户端 423
9.4.5 smbstatus：报告当前 Samba 的连接状态 425
9.4.6 smbmount：装载一个 smbfs 文件系统 426
9.4.7 smbpasswd：设置用户的 SMB 密码 427
9.5 SSH 服务器管理命令 428
9.5.1 sshd：OpenSSH 守护进程 428
9.5.2 ssh-keygen：生成、管理和转换认证密钥 429
9.5.3 ssh：SSH 命令行登录工具 432

9.5.4 sftp：安全互动 FTP......434
9.5.5 scp：将文件复制到远程主机或本地主机......436
9.6 squid：代理服务器 squid 守护进程......437
9.7 DHCP 服务器管理命令......438
9.7.1 dhcpd：DHCP 服务器守护进程......438
9.7.2 dhclient：DHCPv6 客户端守护进程......439
9.7.3 dhcp6c：DHCPv6 客户端守护进程......440
9.8 FTP 服务器管理命令......440
9.8.1 FTP 服务器端管理命令......440
9.8.2 FTP 客户端和 FTP 监控命令......442
9.9 E-mail 服务器管理命令......446
9.9.1 sendmail：为本地或网络交付传送邮件......446
9.9.2 mail：E-mail 管理程序......448
9.9.3 mailq：显示待寄邮件的清单......449
9.9.4 mailstats：显示关于邮件流量的统计信息......450
9.9.5 mutt：电子邮件管理程序......451
9.10 Linux 防火墙管理工具 iptables 和 firewalld......452
9.10.1 iptables：Linux 防火墙管理工具......452
9.10.2 ip6tables：IPv6 版本的 iptables 工具......457
9.10.3 iptables-save：iptables 列表存储......458
9.10.4 iptables-restore：装载由 iptables-save 保存的规则集......458
9.10.5 firewall-cmd：下一代防火墙管理工具......459
9.10.6 arptables：管理 ARP 包过滤的软件......460
9.11 SELinux 管理命令......462
9.11.1 setenforce：设置 SELinux 模式......462
9.11.2 getenforce：查看 SELinux 模式......463
9.11.3 setsebool：设置 SELinux 布尔值......463
9.11.4 getsebool：查看 SELinux 布尔值......463
9.11.5 togglesebool：翻转 SELinux 布尔值......464
9.11.6 sestatus：SELinux 状态查看工具......465
9.11.7 avcstat：显示 AVC 统计信息......466
9.11.8 audit2why：转换审计消息......467
9.11.9 audit2allow：生成策略允许规则......467
9.11.10 load_policy：装载策略......469
9.11.11 semanage：管理 SELinux 策略......469
9.11.12 semodule：管理策略模块......470

9.11.13 chcat：改变语境类别......471
9.11.14 restorecon：恢复文件安全语境......472
9.11.15 chcon：改变文件安全语境......473
9.11.16 setfiles：设置文件安全语境......474
9.11.17 seinfo：提取策略的规则数量统计信息......474
9.11.18 sesearch：搜索 policy.conf 或二进制策略中特别的类型......476
9.11.19 checkmodule：编译策略模块......477
9.11.20 sealert：SELinux 信息诊断客户端工具......477
9.11.21 selinuxenabled：查询系统的 SELinux 是否启用......479
9.11.22 SELinux 工具总结列表......479
9.12 虚拟化管理命令......481
9.12.1 xen 虚拟化管理命令......481
9.12.2 KVM/Qemu 虚拟机管理命令......495

第 10 章 Linux 打印管理命令......504

10.1 cupsd：通用打印程序守护进程......504
10.2 cupsaccept：指示打印系统接受发往指定目标打印机的打印任务......505
10.3 lpadmin：配置 LP 打印服务......506
10.4 lp：打印文件......508
10.5 lpstat：显示行式打印机的状态信息......509
10.6 lpr：排队打印作业......510
10.7 lprm：从打印队列中删除任务......511
10.8 lpc：控制打印机......512
10.9 lpq：检查假脱机队列......513
10.10 lpinfo：显示驱动和设备......514
10.11 lpmove：将作业从一个队列移动到另一个队列......515
10.12 cancel：取消已存在的打印任务......516
10.13 cupsdisable：禁用指定的打印机或类......516
10.14 cupsreject：指示打印系统拒绝发往指定目标打印机的打印任务......517
10.15 cupsenable：启动指定的打印机......517

第 11 章 Linux 库应用命令......518

11.1 ldconfig：配置查找共享库......518
11.2 nm：列举目标文件中的符号名......519
11.3 ar：建立修改文件或从文件中抽取成员......522
11.4 strip：去除目标文件中的无用信息......523
11.5 objdump：展开目标文件、静态库和共享库中的信息......524

11.6　ldd：显示共享库的依赖情况 526
11.7　patch 命令：修补文件 527
11.8　ld：连接器 529

第 12 章　Linux 开发应用命令 532

12.1　as：标准 GNU 汇编程序 532
12.2　nasm：汇编器 534
12.3　gcc：C/C++编译器 535
12.4　make：维护和编译软件或软件包 543
12.5　gdb：GUN 调试器 547
12.6　gdbserver：远端 GNU 服务器 554
12.7　autoconf：产生配置脚本 555
12.8　autoheader：为 configure 产生模板头文件 555
12.9　autoreconf：更新已经生成的配置文件 556
12.10　autoscan：产生初步的 configure.in 文件 556
12.11　autoupdate：更新 configure.in 文件 557
12.12　automake：自动生成 Makefile.in 的工具 557
12.13　aclocal：生成 aclocal.m4 文件 558
12.14　configure：生成 Makefile 文件 558

第 13 章　Linux 行编辑器 561

13.1　ed：文本编辑器 561
13.2　sed：利用 script 命令处理文本文件 562
13.3　pico：编辑文字文件 564
13.4　vim：全屏幕文本编辑器 566
13.5　ex：文本编辑器 572
13.6　nano：文本编辑器 572
13.7　jed：编辑文本文件 575
13.8　joe：编辑文本文件 576

第 14 章　Linux Shell 命令 578

14.1　Shell 简介 578
14.1.1　什么是 Shell 578
14.1.2　Shell 类型 579
14.2　Linux Shell 的功能 580
14.2.1　通配符 580
14.2.2　重定向 580

14.2.3 管道 582
14.2.4 别名 583
14.2.5 命令行补全 583
14.2.6 命令替换 583
14.2.7 命令历史 584
14.3 Linux Shell 元字符 587
14.4 Shell 变量和 Shell 环境 588
14.4.1 变量简介 588
14.4.2 了解 Shell 的内置命令 589

01 Linux 命令行简介

Linux 操作系统是目前较为流行的操作系统之一，相对于其他操作系统而言，Linux 操作系统具有独特的特性和优点。为了帮助读者对 Linux 操作系统有一个初步的认识，本章将对 Linux 的命令行与基本操作进行简单的介绍。

1.1 Linux 命令行概述

1.1.1 Linux 命令行的重要作用

Linux 命令行是 Linux 系统的核心，利用命令行可以对系统进行各种配置。要熟练并成功地管理 Linux 操作系统，必须对 Linux 下的命令行有深入的理解。Linux 下的命令行有助于初学者了解系统的运行情况和计算机的各种设备，如中央处理器、内存、磁盘驱动、键盘、鼠标及其他输入/输出设备和用户文件都是在 Linux 系统管理命令下运行的。可以说，Linux 的命令行对于整个系统的运行，以及设备与文件之间的协调具有核心的作用。

虽然近几年个人计算机的潮流已经从命令行的方式向图形界面的方式转换，用户也热衷于利用鼠标在图形界面下对系统进行操作，但是命令行仍然有其强大的生命力，如系统管理员必须经常和命令行打交道。与图形界面相比，利用命令行对系统进行管理和操作显得更为快捷与安全。特别是在服务器应用方面，Linux 命令行更是一种核心的应用，这主要是因为在字符界面下的操作具有更好的稳定性和安全性。

1.1.2 Linux 命令行介绍

在 Linux 操作系统中，命令行处于核心的地位。命令行是一种对操作系统的输入和输出界面，与图形界面对应。目前，在计算机操作系统中图形界面成为主流。然而，作为字符界面的命令行由于具有占用系统资源少、性能稳定并且非常安全等特点仍发挥着重要作用，Linux 命令行在服务器中一直有着广泛应用。利用命令行可以对系统进行各种操作，这些操作虽然没有图形化界面那样直观明了，但是却显得快捷而顺畅。在 Linux 中，命令行命令有大小写的区分，且所有的 Linux 命令行命令和选项都区分大小写，如-V 和-v 是两个不同的命令，这与 Windows 操作系统有所区别。在 Windows 操作系统环境下，所有的命令都没有大小写的区别。初学者应遵循“所有控制台命令的输入均小写”这一原则。例如查看当前目录文件，在命令行下输入：

```
# ls
anaconda-ks.cfg  Desktop  install.log  install.log.syslog  local
scsrun.log
```

若在命令行下输入：

```
# Ls
-bash: Ls: command not found
```

系统将给出命令错误的信息“命令未找到”。

1.1.3　Linux 命令行的启动及退出

启动桌面发行版的 Linux 系统，进入图形界面后，可以通过 init 3 命令回到命令行登录界面，按 Alt+Fn（*n*=1,2,3,⋯,6）组合键进入字符界面，例如按 Alt+F3 组合键即可进入虚拟控制台 tty3（tty7 通常运行 X Window 的图形界面），此时输入正确的用户名及登录密码后即可进行字符界面的命令行操作，如图 1-1 所示。

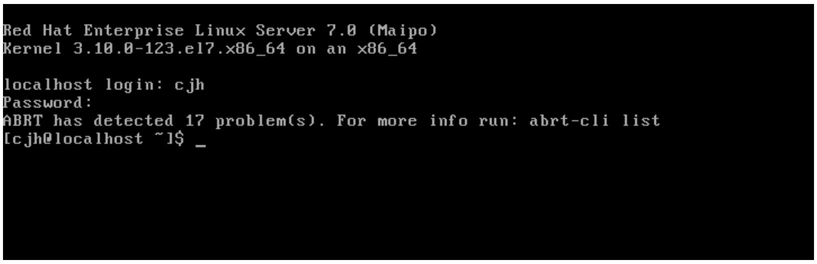


图 1-1　字符界面命令行登录系统

在这个示例中，第一次输入的是一个错误的密码，系统提示登录不正确并提示重新登录，直到输入了正确的用户名和登录密码后，登录成功。注意，Linux 系统不会显示输入的密码，而且不论是输入的用户名错误还是输入的密码错误，系统都将给出同样的错误信息并提示重新输入用户名和密码。

如果要退出 Linux 命令行，可在命令提示符后面输入命令 exit 或按 Ctrl+D 组合键。例如，在上例中的命令提示符“$”后输入 exit，然后按 Enter 键或 Ctrl+D 组合键，即可退出登录的系统。

1.1.4　Linux 命令行提示符

Linux 命令行下的提示符有两种：“$”和“#”。通常登录到命令行后看到的提示符为美元符号“$”，如图 1-1 所示。在示例中最后一行的“$”符号即为 Linux 的一种命令提示符（也称 Shell 提示符），代表当前用户具有普通用户权限，cjh 为登录的用户名，“@”后面的 localhost 为主机名，主机名后是当前的工作目录，代字符“～”表示用户的主目录，即当前工作目录为用户主目录。若当前工作目录为主目录下的 tmp 目录，则“～”将被“tmp”所代替。通过 pwd 命令即可显示当前工作目录的绝对路径，如下：

```
$ pwd
/home/cjh
```

如果当前用户是超级用户 root（即以用户名 root 登录系统），将会看到另一种命令行提示符“#”，表示用户拥有最高权限，可对系统做任意修改，如图 1-2 所示。

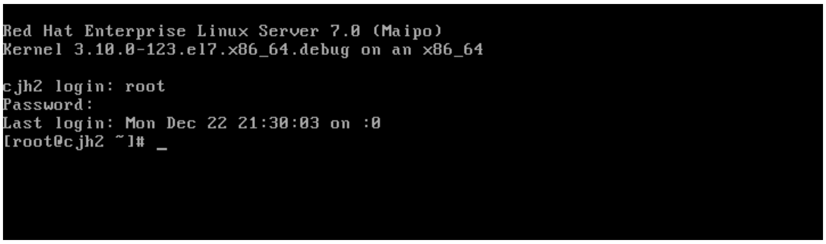


图 1-2　root 用户命令提示符

从示例中可以看出，此处的代字符“～”代替的路径是“/root”，而上一例中 “～”代替的路径是“/home/cjh”，即代字符“～”代替的路径始终是当前用户的主目录，其绝对路径名随登录用户的不同而改变。

1.1.5　命令行的历史记录和编辑

目前的 Linux 发行版默认使用 Bash Shell 命令行，它已成为 Linux 系统的实际标准。Bash Shell 命令行具有非常强大的功能，它提供的一些特性使输入命令变得更加容易。

1．命令行的历史记录

可以通过重复按向上方向键↑遍历近来在控制台下输入的命令，按向下方向键↓可以向前遍历，与 Shift 键联用可以遍历以往在该控制台中的输出。例如可以在命令行下重复按向上方向键↑，直到出现近来在控制台下输入的命令 ls 为止，然后按 Enter 键执行这一命令。

2．编辑命令行

在命令行下按 Ctrl+R 组合键将进入向后增量搜索模式，命令行界面出现如下命令：

```
(reverse-i-search)`l': ls
```

此时尝试输入以前输入过的命令，每输入一个字符，命令行都会滚动显示历史命令。当显示到想要查找的命令时，直接按 Enter 即可执行该历史命令。仍以 ls 命令为例，当输入字符 l 时，搜索出来的命令是所需要的历史命令 ls，然后按 Enter 键即执行该命令。

在命令行下按 Ctrl+P 或者 Ctrl+N 组合键可以快速向前或向后滚动查找一个历史命令，这样可以快速提取刚刚执行过不久的命令。例如，在命令行下输入如下命令：

```
echo "Hello，Linux world"
```

命令行下将出现字符“Hello，Linux world”，然后按 Ctrl+P 组合键，命令行下将出现刚刚输入过的命令。

1.1.6 命令行快捷键

通过一些功能键可以快速浏览并编辑命令行。下面给出常用的完成一般编辑的快捷键，如表 1-1 所示。

表 1-1 Linux 命令行快捷键

快 捷 键	说 明
↑ 或 Ctrl+P	显示上一条命令
↓ 或 Ctrl+N	显示下一条命令
→ 或 Ctrl+F	光标向右移动一个字符
← 或 Ctrl+B	光标向左移动一个字符
Alt+F	光标向右移动一个单词
Alt+B	光标向左移动一个单词
Ctrl+R	逆向搜索包含输入字符串的命令，继续按 Ctrl+R 组合键，再向上搜索，可使用 Ctrl+C 组合键中断命令
Ctrl+A	光标移动到开头
Ctrl+E	光标移动到结尾
Ctrl+L	清屏
Ctrl+U	剪切光标所在处之前的所有字符（不包括自身）
Ctrl+K	剪切光标所在处之后的所有字符（包括自身）
Ctrl+W	剪切光标所在处之前的一个词（以空格、标点等为分隔符）
Ctrl+Y	粘贴
Alt+D	删除从光标当前位置到当前字的结尾字符
Alt+U	将光标所在处的单词转换为大写（从光标处到词的结尾）
Alt+L	将光标所在处的单词转换为小写（从光标处到词的结尾）
Alt+C	将光标所在处的单词转换为首字母大写（从光标处到词的结尾）
Ctrl+C	中断命令
Ctrl+S	挂起当前 Shell
Ctrl+Q	重新启用挂起的 Shell
Ctrl+Z	暂停命令
!$	重复前一个命令最后的参数
Ctrl+(X U)	按住 Ctrl 键的同时先后按 X 键和 U 键，撤销刚才的操作
Esc+T	颠倒光标所在处及其相邻单词的位置

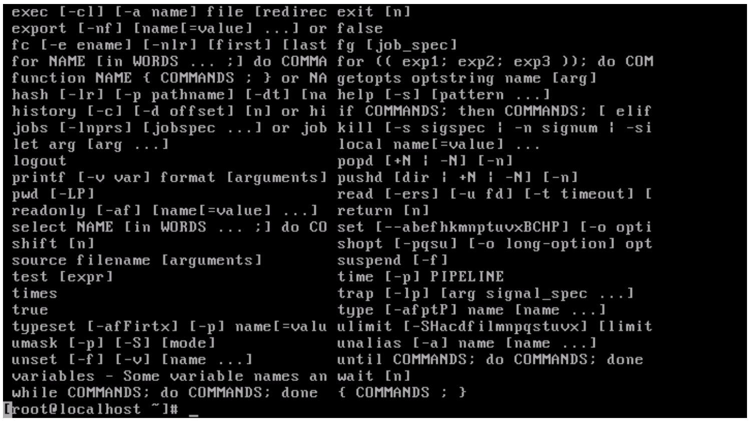
1.2 在命令行下使用 Linux 帮助信息

许多初学者安装完 Linux 操作系统后不知道从哪里入手学习 Linux，特别是感觉到没有帮

助信息。Linux 发行版本对用户提供的技术支持时间比较短。其实在安装了一个完整的 Linux 系统后，其中已经包含了一个强大的帮助系统，只是可能用户还没有发现和使用它的技巧。下面以 RHEL 5 为例，帮助读者深入挖掘隐藏在 Linux 系统中的帮助信息。

1.2.1　使用 help 命令获得 bash 的内部命令帮助

bash 的内部命令有 40 个，主要包括 exit、less、lp、kill、cd、pwd、fc、fg 等。bash 内置的命令列表 A——通过在命令行输入 help 即可获得。这些内部命令没有独立的命令程序（即无法搜索到这些命令）和帮助文件，help 命令提供这些命令的在线帮助。有趣的是，help 命令本身也是内部命令。下面使用 help 命令获取它提供帮助命令的列表，如图 1-3 所示。



```
exec [-cl] [-a name] file [redirec exit [n]
export [-nf] [name[=value] ...] or false
fc [-e ename] [-nlr] [first] [last fg [job_spec]
for NAME [in WORDS ... ;] do COMMA for (( exp1; exp2; exp3 )); do COM
function NAME { COMMANDS ; } or NA getopts optstring name [arg]
hash [-lr] [-p pathname] [-dt] [na help [-s] [pattern ...]
history [-c] [-d offset] [n] or hi if COMMANDS; then COMMANDS; [ elif
jobs [-lnprs] [jobspec ...] or job kill [-s sigspec | -n signum | -si
let arg [arg ...]                  local name[=value] ...
logout                             popd [+N | -N] [-n]
printf [-v var] format [arguments] pushd [dir | +N | -N] [-n]
pwd [-LP]                          read [-ers] [-u fd] [-t timeout] [
readonly [-af] [name[=value] ...]  return [n]
select NAME [in WORDS ... ;] do CO set [--abefhkmnptuvxBCHP] [-o opti
shift [n]                          shopt [-pqsu] [-o long-option] opt
source filename [arguments]        suspend [-f]
test [expr]                        time [-p] PIPELINE
times                              trap [-lp] [arg signal_spec ...]
true                               type [-afptP] name [name ...]
typeset [-afFirtx] [-p] name[=valu ulimit [-SHacdfilmnpqstuvx] [limit
umask [-p] [-S] [mode]             unalias [-a] name [name ...]
unset [-f] [-v] [name ...]         until COMMANDS; do COMMANDS; done
variables - Some variable names an wait [n]
while COMMANDS; do COMMANDS; done  { COMMANDS ; }
[root@localhost ~]# _
```

图 1-3　help 命令列表

在这个示例中，由于 help 命令列表较长而不能在一页内全部显示，所以能看到的列表其实是 help 命令列表的最后一页。与 MS-DOS 一样，要想看到该命令的所有帮助内容，需要在命令后添加一个选项参数。可以在命令提示符下输入命令：

```
help |more
```

即可看到 help 命令列表的第一页，再次按 Enter 键将一行行向后翻页，按 Space 键将直接跳转到最后一页。若想在中途退出帮助文档，可以按 Q 键，这样将直接退出文档，回到命令提示符下。

使用 help 命令提供某命令的帮助方法非常简单，在 help 命令后输入空格和命令名称即可。例如：

```
$ help kill
```

help 命令提供的 kill 命令参数很详细，当用户对 Linux 有一定了解后往往只需要主要的语法，那么可以使用“-s”参数。

```
$ help -s kill
kill: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill
-l [sigspec]
```

1.2.2 使用 man 获取帮助信息

1．man 简介和使用

在文本窗口中浏览文件的方式在普遍使用 HTML 和 PDF 的时代似乎有些过时了，但在终端窗口中阅读文件仍有其优势。对于那些 Linux 开发者来说，这种方式是可取的，因为大部分工作是在终端窗口中完成的。从 man 页面上获取信息比打开浏览器或 PDF 浏览器要快得多。man 页面具有简明的格式，这就意味着它采用线性格式来显示。简单的工具通过一个 man 页面就可以获取，作为它们唯一的文件来源；较为复杂的工具和程序库依赖于合适的 man 页面作为文档。创建 man 页面的工具可以选择更多的字符编码，例如 UTF-8，所以现在 man 页面可以用任何语言来编写。

使用过 UNIX 的用户比较熟悉 man 手册页。man 是 manual（手册）的缩写，是在 Linux 系统中存储的一部联机使用的手册，以供用户在终端上查找。例如，如果使用 ps 命令时遇到困难，可以输入 man ps 得到帮助信息，这时会显示出 ps 的手册页，如图 1-4 所示。

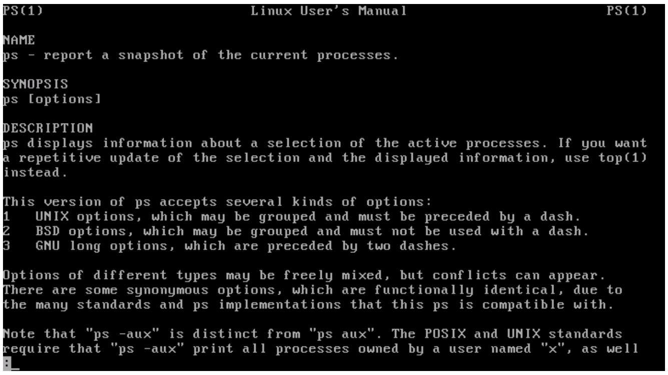


图 1-4 man 页面

手册页 man 是用 less 程序来看的（可以方便地使屏幕上翻和下翻），所以在 man 页面中可以使用 less 的所有选项。在 less 中比较重要的功能键有：[q]退出，[Enter]一行行地下翻，[Space]一页页地下翻，[b]上翻一页，[/]后跟一个字符串，[Enter]查找字符串，[n]发现上一次查找的下一个匹配。手册页（man page）在很小的空间里提供了大量的信息。大多数手册页中都有的部分，即 Linux 手册页的主要部分有以下几个。

- 1：用户命令的使用方法、可以使用的参数等。
- 2：系统调用，只有系统才能执行的函数。
- 3：库调用，大多是 libc 函数，如 qsort(3)。
- 4：设备和特殊文件。
- 5：文件格式和约定。
- 6：游戏。
- 7：宏命令包和约定。
- 8：系统管理命令，多数只有 root 可以执行。
- 9：内核程序。
- n：Tcl 或 Tk 指令。

man 手册页文件存放在/usr/share/man 目录下。Linux 手册页布局如下。

– NAME：程序或者命令的名称、手册节号及发布日期。
– SYNOPSIS：怎样调用命令，带有所有选项和参数的完整列表。
– DESCRIPTl0N：命令及其用法的简短小结。
– RETURN VALUES：程序或者库函数返回值，以及产生特定返回值的环境。
– EXIT STATUS：经常用来替代 TURNVALUS。
– OPTIONS：按字母顺序排列的选项和参数清单（如果有的话）。
– FILES：命令使用的或者能使用的文件清单。
– USAGE：用程序的语言说明的简明语法（如果有的话）。
– ENVIROMENT：命令使用的或者能使用的环境变量清单。
– DIAGNOSTICS：命令产生的错误信息及其解决办法的清单。
– NOTES：不能归入其他任何一种类别下的所有信息。
– CONFORMING TO：列出程序遵循的任何标准，比如 POSIX 或 ISO。
– SEE ALSO：与命令有关的交叉索引和信息。
– BUGS：指出已知的 bug 和错误功能，以及怎样和程序的作者联系修正它们。
– AUTHOR：命令的作者或者维护者的名字，可能带有电子邮件地址或 URL 地址。

说明

在 Linux 系统中 man 存放在不同的 man 文件夹下，例如 man1、man2 等，每部分都描述了相关工具的使用方法，最常用的是 man1（用户命令）、man5（文件格式）和 man8（系统管理）。有些情况下，不同工具的手册对应相同的名字。例如，在命令行输入 man clear 可以查看 clear 程序的 man 页，而输入 man 3 clear 则可以查看 clear 子程序的 man 页。

2．汉化 man

可以下载相关软件包把手册页汉化，过程如下：

```
#wegt http://download.sf.linuxforum.net/cmpp/man-pages-zh_CN-1.5.tar.gz
#gunzip man-pages-zh_CN-1.5.tar.gz
#tar vxf man-pages-zh_CN-1.5.tar
#cd man-pages-zh_CN-1.5
```

安装之前，运行 locale 命令查看自己的语言环境是什么，主要有影响的是 LC_ALL、LC_CTYPE 和 LANG 变量的值。如果语言环境是 zh_CN.UTF-8，可以运行 make u8 && make install-u8；如果语言环境是 GB2312、GBK 或者 GB18030，可以运行 make gb && make install-gb；或者可以下载对应的 rpm 包来安装，同样简单。对于 zh_CN.UTF-8，应当安装 man-pages-zh_CN-1.x；对于 GB2312、GBK、GB18030，应当安装 man-pages-zh_CN-gb-1.x（请参阅讨论版上关于不同发行版的系统与其默认中文编码的说明）。在安装 man-pages-zh_CN-gb-1.x 之后，应当注销退出，在重新登录之后才能继续使用手册页系统；而在安装适于 UTF-8 语言环境的

man-pages-zh_CN-1.x 之后，不必注销就可以使用。使用方法也有所不同，对于 GB*语言环境的系统，应当使用 cman 命令来查看，例如“cman ls”；对于 UTF-8 语言环境的系统，只要使用 man 命令就可以了。如图 1-5 所示为 man 命令本身的汉化手册页。

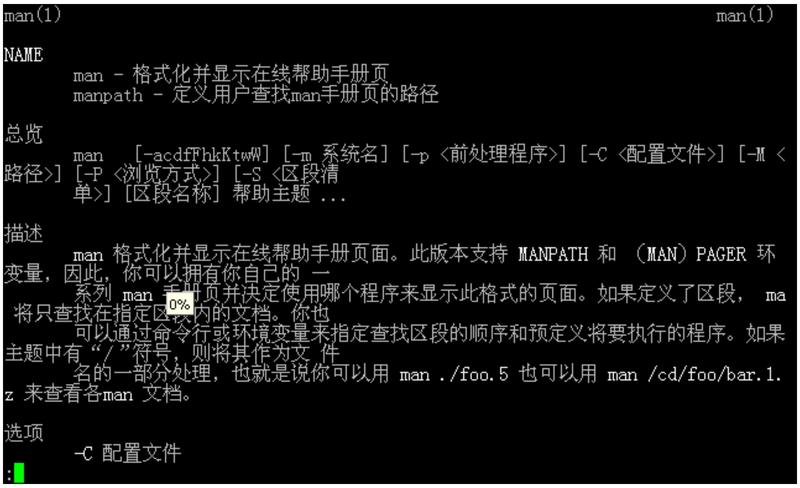


图 1-5　man 命令本身的汉化手册页

1.2.3　使用 info 获取帮助信息

info 是 Linux 系统提供的另外一种格式的帮助信息。和 man 相比，info 具有更好的交互功能，它支持链接跳转。通常使用 info 和 pinfo 命令来阅读 info 文档。Linux 中的大多数软件开发工具都来自自由软件基金会的 GNU 项目，这些软件的在线文档都以 info 文件的形式存在。info 程序是 GNU 的超文本帮助系统。

1．运行 info

可以在 Shell 提示符后输入 info（不要参数），它将列出一个文档清单，如图 1-6 所示。

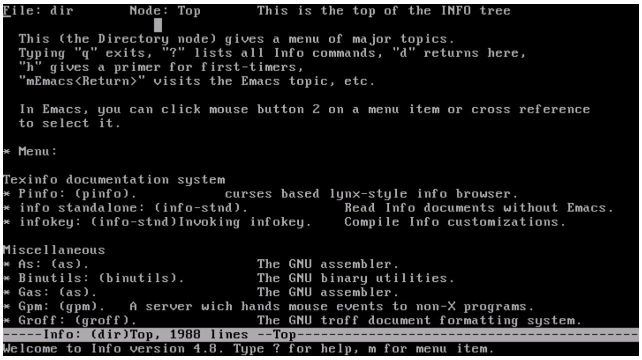


图 1-6　info 文档的分类列表

如果没有发现所需要的，那是因为没有安装包含那个文档的软件包，安装后再试。info 帮助系统的初始屏幕将显示一个主题目录，可以将光标移动到带有*的主题菜单上面，然后按 Enter

键进入该主题；也可以输入 m，后跟主题菜单的名称进入该主题。例如，输入 m，然后再输入 gcc 就会进入 gcc 主题中。info 系统是一个超文本系统，任何高亮显示的文字都有一个链接导向更多的信息。按 Tab 键将光标移到链接处，并按 Enter 键进入链接，按 p 键返回上一页，按 n 键翻到下一页，按 u 键回到文档的上一层。用 C-h 获得帮助，用 m 进入菜单界面。如果要在主题之间频繁跳转，记住如下几个命令键。

- n：跳转到该节点的下一个节点。
- p：跳转到该节点的上一个节点。
- m：指定菜单名而选择另外一个节点。
- f：进入交叉引用主题。
- l：进入该窗口中的最后一个节点。
- Tab：跳转到该窗口的下一个超文本链接。
- RET：进入光标处的超文本链接。
- u：转到上一级主题。
- d：回到 info 的初始节点目录。
- h：调出 info 教程。
- q：退出 info。

2．使用 pinfo

pinfo 兼容 info 的功能，并且支持彩色链接文本、鼠标选定支持等功能。运行 pinfo，代码如下：

```
#pinfo
```

pinfo 提供的基于浏览器风格的 info 文档程序界面如图 1-7 所示，使得操作更加简单、界面更加友好。

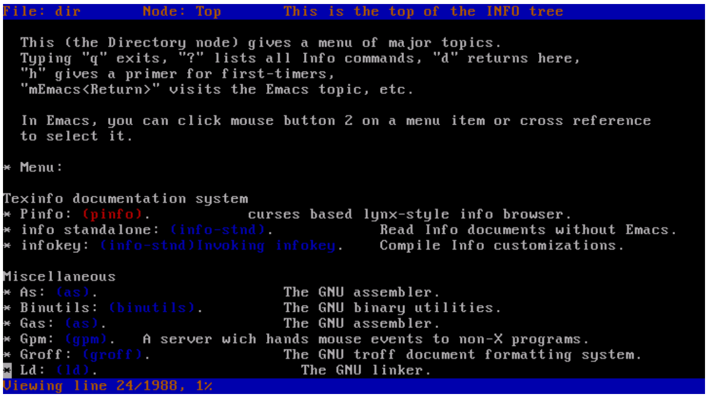


图 1-7　pinfo 使用界面

pinfo 快捷键定义在/etc/pinforc 文档中，这些快捷键可以更改。常用的快捷键如下。

- s：该帮助的所有节点查询。
- /：该帮助的当前节点查询。

– h：父节点。
– j：文档下翻一行。
– k：文档上翻一行。
– Space：下一页。
– e：最后一页。
– H：最前一页。
– d info：所有指令的目录。

1.2.4　用发行版的系统手册

RHEL 7.0 作为出色的发行版本提供了一套优秀的文档：发行版系统手册。可以从 RedHat 的官方网站下载该手册。下载网址为 https://access.redhat.com/documentation/zh-CN/Red_Hat_Enterprise_Linux/7/。RHEL 7.0 手册页如表 1-2 所示。

表 1-2　RHEL 7.0 手册页

手册内容	文件名称	语　言	格　　式
发行注记	RELEASE-NOTES-x86-zh_CN	中文	.pdf，.html，.epu
部署指南	Deployment_Guide	中文	.pdf，.html，.epu
安装指南	Installation_Guide	中文	.pdf，.html，.epu
安全指南	Security_Guide	中文	.pdf，.html，.epu
迁移规划指南	Migration_Planning_Guide	中文	.pdf，.html，.epu
性能调节指南	Performance_Tuning_Guide	中文	.pdf，.html，.epu
虚拟化指南	Virtualization	英文	.pdf，.html，.epu
系统管理指南	System Administrator's Guide	英文	.pdf，.html，.epu
网络指南	Networking Guide	英文	.pdf，.html，.epu
群集器管理	Cluster_Administration	英文	.pdf，.html，.epu
全球网络块设备	Global_Network_Block_Device	英文	.pdf，.html，.epu
SELinux 使用指南	SELinux User's and Administrator's Guide	英文	.pdf，.html，.epu
负载均衡指南	Load Balancer Administration	英文	.pdf，.html，.epu
群集器逻辑卷管理	Cluster_Logical_Volume_Manager	英文	.pdf，.html，.epu
虚拟服务器管理	Virtual_Server_Administration	英文	.pdf，.html，.epu

1.2.5　从 Internet 获得帮助

Internet 上提供了许多 Linux 方面的站点及相关文档，除此以外，还可以利用搜索引擎，如百度（www.baidu.com）、Google（www.google.com），输入遇到的错误信息进行搜索，通常可以找到问题的相关介绍和解决办法。

1．Linux 文档项目

Linux 文档项目（Linux Documentation Project，LDP）提供了大量关于 Linux 的免费发布书

籍，其历史几乎和 Linux 一样长，内容包括使用指南、FAQ、HOWTO、杂志、man 页等。LDP 的主页是 www.tldp.org，主页支持多种语言，可以进行本地搜索，简单易用。提供的文档类型如下。

- HOWTOs 文档：HOWTOs 文档采用多种格式发行，如.txt、.html、.pdf 等流行的文档格式都有，读者可以根据需要下载相应的文档压缩包进行离线阅读。HOWTOs 的作者会根据需要不定期地对文档进行版本更新，通常文档的更新是跟随某个相关软件或发行版本的升级而进行的。
- mini－HOWTOs 文档：mini－HOWTOs 文档内容和风格与 HOWTOs 文档非常相似，只是文档的主题和篇幅要比 HOWTOs 文档小一些。由于其主题小而对于解决某个问题更有针对性，多为某个类型问题的使用技巧。通过 mini－HOWTOs 文档通常可以快速解决现有问题。
- man 手册页：和联机手册页相比，LDP 的 man 手册页会定期进行更新，读者可以在 LDP 的站点获得最新的命令手册页。
- FAQ 文档：FAQ 文档采用一问一答的形式对常见的问题给出解决方法，往往能够在找到问题之后快速解决问题（见 http://www.tldp.org/FAQ/LDP-FAQ/）。
- Guide 文档：Guide 文档是对 Linux 某个方面应用的指南，主体范围较大、篇幅较长，读者可以把它作为比较系统的教程来阅读（见 http://www.tldp.org/guides.html）。

2．GNU

在 GNU（GNU’s Not UNIX）的主页（www.gnu.org）上可以得到很多 GNU 的文档及其他资源。与 LDP 一样，GNU 的文档也提供多语言版本，在 www.gnu.org/manual 站点可以获得 GNU 的手册页。

3．各 Linux 发行版官方网站及 BBS 论坛

大多数 Linux 发行版都有其主页和 BBS，上面提供了许多文档，若有问题也可以在 BBS 上留言寻求帮助。其中 Debian（www.debian.org）、Ubuntu（www.ubuntu.org）、Gentoo（www.gentoo.com）和 RedHat 的网站上都提供了非常优秀的材料。此外还可以在一些非常优秀的 BBS 上寻求帮助，如中国 Linux 论坛（www.linuxforum.net）、ChinaUnix（www.chinaunix.net）、www.linuxsir.org 等。

02 Linux 文件管理命令

2.1 ls：显示文件名

作用：ls 命令用于显示目录内容，类似于 DOS 下的 dir 命令，它的使用权限是所有用户。

用法：ls [选项]...[文件]...

主要选项如下。

- -a,--all：不隐藏任何以“.”开始的项目。
- -A,--almost-all：列出除“.”及“…”以外的任何项目。
- --author：与-l 同时使用时列出每个文件的作者。
- -b,--escape：以八进制溢出序列表示不可打印的字符。
- -B,--ignore-backups：不列出任何以“~”字符结束的项目。
- -c：与“-lt”选项连用时，按照文件状态时间排序输出目录内容，排序的依据是文件的索引节点中的 ctime 字段；与“-l”选项连用时，则排序的依据是文件的状态改变时间。
- -C：每栏由上至下列出项目。
- --color[=WHEN]：控制是否使用色彩分辨文件。WHEN 可以是“never”（默认）、“always”或“auto”其中之一。
- -d,--directory：当遇到目录时列出目录本身而非目录内的文件。
- -D,--dired：产生适合 Emacs 的 dired 模式使用的结果。
- -f：不进行排序，-aU 选项生效，-lst 选项失效。
- -F,--classify：加上文件类型的指示符号（*、/、=、@、|其中一个）。
- --format=关键字：交错-x，逗号分隔-m，水平-x，长-l，单栏-1，详细-l，垂直-C。
- --full-time：即-l--time-style=full-iso。
- -g：类似-l，但不列出所有者。
- --group-directories-first：在文件前分组目录。此选项可与--sort 一起使用，但是一旦使用--sort=none(-U)将禁用分组。
- -G,--no-group：以一个长列表的形式，不输出组名。
- -h,--human-readable：与-l 一起，以易于阅读的格式输出文件大小（如 1KB、234MB、2GB）。

- --si：同上面类似，但是使用 1000 为基底而非 1024。
- -H,--dereference-command-line：跟随命令行列出的符号链接。
- --dereference-command-line-symlink-to-dir：跟随命令行列出的目录的符号链接。
- --hide=PATTERN：隐藏符合 PATTERN 模式的项目（-a 或-A 将覆盖此选项）。
- --indicator-style=方式：指定在每个项目名称后加上指示符号方式，none（默认），classify（-F），file-type（-p）。
- -i,--inode：显示每个文件的 inode 号。
- -I,--ignore=PATTERN：不显示任何符合指定 shellPATTERN 的项目。
- -k：即--block-size=1K。
- -l：使用较长格式列出信息。
- -L,--dereference：当显示符号链接的文件信息时，显示符号链接所指示的对象而并非符号链接本身的信息。
- -m：所有项目以逗号分隔，并填满整行行宽。
- -n,--numeric-uid-gid：类似-l，但列出 UID 及 GID。
- -N,--literal：输出未经处理的项目名称（如不特别处理控制字符）。
- -o：类似-l，但不列出有关组的信息。
- -p,--indicator-style=slash：给目录加上表示符号“/”。
- -q,--hide-control-chars：以“?”字符代替无法打印的字符。
- --show-control-chars：直接显示无法打印的字符（这是默认方式，除非调用的程序名称是“ls”，而且是在终端输出结果）。
- -Q,--quote-name：将条目名称括上双引号。
- --quoting-style=方式：使用指定的 quoting 方式显示条目的名称，如 literal、locale、shell、shell-always、c、escape。
- -r,--reverse：逆序排列。
- -R,--recursive：递归显示子目录。
- -s,--size：以块数形式显示每个文件分配的尺寸。
- -S：根据文件大小排序。
- --sort=WORD：以下是可选用的 WORD 和它们代表的相应选项。

```
extension    -X   status   -c
none         -U   time     -t
size         -S   atime    -u
time         -t   access   -u
version      -v   use      -u
```

- --time=WORD：和-l 同时使用时显示 WORD 所代表的时间而非修改时间，如 atime、access、use、ctime 或 status；加上--sort=time 选项时会以指定时间作为排序关键字。
- --time-style=STYLE：和-l 同时使用时根据 STYLE 代表的格式显示时间，如 full-iso、iso、locale、posix-iso、+FORMAT。FORMAT 即是“date”所用的时间格式；如果 FORMAT 是 FORMAT1<换行>FORMAT2，则 FORMAT1 适用于较旧的文件而 FORMAT2 适用于

较新的文件；如果 STYLE 以“posix-”开头，则 STYLE 仅在 POSIX 语系之外生效。

- -t：根据修改时间排序。
- -T,--tabsize=宽度：指定制表符（Tab）的宽度，而非 8 个字符。
- -T,--tabsize=COLS：指定制表符（Tab）的 COLS 数值，而非 8 个字符。
- -u：同-lt 一起使用：按照访问时间排序并显示；同-l 一起使用，显示访问时间并按文件名排序；其他，按照访问时间排序。
- -U：不进行排序；按照目录顺序列出项目。
- -v：在文本中进行数字（版本）的自然排序。
- -w,--width=COLS：自行指定屏幕宽度而不使用目前的数值。
- -x：逐行列出项目而不是逐栏列出。
- -X：根据扩展名排序。
- -1：每行只列出一个文件。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

SELinux 选项如下。

- -Z：显示安全上下文和用户、用户组相关文件。
- --scontext：显示安全上下文和相关文件。

SIZE 可以是一个可选的整数，后面跟着以下单位中的一个。KB：1000，K：1024，MB：1000×1000，M：1024×1024，以及 G、T、P、E、Z、Y。

使用色彩来区分文件类型的功能已被禁用，默认设置和--color=never 同时禁用了它。

使用--color=auto 选项，ls 只在标准输出被连至终端时才生成颜色代码。

LS_COLORS 环境变量可改变此设置，可使用 dircolors 命令来设置。

退出状态：

```
0 正常
1 一般问题（例如，无法访问子文件夹）
2 严重问题（例如，无法使用命令行参数）
```

应用实例如下。

（1）ls 命令的颜色选项。

ls 命令是 Linux 系统中使用频率最高的命令，它的参数也是 Linux 命令中最多的。使用 ls 命令时会有几种不同的颜色，其中蓝色表示目录，绿色表示可执行文件，红色表示压缩文件，浅蓝色表示链接文件，加粗的黑色表示符号链接，灰色表示其他用法文件。ls 命令最常使用的参数是 ls -l。

（2）使用窍门。

要想看到 ls 命令的所有选项，可以通过在 Shell 提示下使用命令 man ls 来阅读其说明书页。要想打印这个说明书页，可以在 Shell 提示下使用命令 man ls | col -b | lpr。

（3）使用命令 ls –a。

执行该命令后将会看到以点开始的文件。

```
[cjh1@localhost ~]$ ls -a
.  ..  .bash_logout  .bash_profile  .bashrc  .mozilla
```

隐藏文件多数是配置文件，它们给程序、窗口管理器、Shell 等设置首选项。它们被隐藏的目的是防止用户对其无意的篡改。当在目录中搜寻某项事物时，一般不是在寻找这些配置文件，因而在 Shell 下查看目录内容时把它们隐藏起来可以避免屏幕的拥挤。使用 ls -a 命令查看所有的文件会显示大量的细节，但是通过添加更多的选项，可以看到更多的细节。如果想查看一个文件或目录的大小、创建时间等，在 ls -a 命令后面添加 long（长）选项（-l）就可以了。这个命令显示了文件创建的日期、大小、所有者、权限等。在使用 ls 命令查看目录内容时，不必位于该目录下。例如，要在主目录中查看/usr 目录中的内容，应使用如下命令：

```
[cjh1@localhost ~]$ ls -al /usr
总计 272
drwxr-xr-x  14 root root  4096 2008-10-30 .
drwxr-xr-x  23 root root  4096 03-23 10:31 ..
drwxr-xr-x   2 root root 65536 03-04 12:59 bin
…
```

（4）查看文件被修改或被访问的时间。

在查看文件的属性时会发现它的时间标记，比如下面示例中的 03-23 11:11。这个时间并不代表文件被创建的时间，而是代表文件被访问或被修改的时间。文件被修改的时间比较容易理解，比如我们可以用编辑器来修改文本文件，然后保存一下，这样文件的时间就改变了。示例代码如下：

```
[cjh1@localhost ~]$ ls -l adduml02.sh
-rw-rw-r-- 1 cjh1 cjh1 0 03-23 11:11 adduml02.sh
```

（5）查看当前 Linux 支持的文件类型。

不同版本的 Linux 所支持的文件系统类型和种类都有所不同，如何知道自己使用的 Linux 发行版的文件系统类型呢？下面以 RHEL 7 为例，讲解如何操作。以超级用户权限登录 Linux，进入/lib/modules/3.10.0-123.el7.x86_64/kernel/fs/目录，执行下面的命令：

```
# cd /lib/modules/3.10.0-123.el7.x86_64/kernel/fs/
# ls
autofs4     cramfs    ext3     fscache  hfsplus  lockd       nfsd     vfat
cachefiles  dlm       ext4     fuse     jbd      msdos       nls
cifs        ecryptfs  fat      gfs2     jbd2     nfs         squashfs
configfs    exportfs  freevxfs hfs      jffs2    nfs_common  udf
```

说明

不同 Linux 发行版本的 fs 目录的位置可能有所不同，其中显示了当前系统所支持的文件系统种类。可以看到，RHEL 7 支持的文件系统非常多。

2.2 cat：显示文本文件内容

作用：将[文件]或标准输入组合输出到标准输出。

用法：cat [选项] [文件]...

主要选项如下。

- -A,--show-all：等价于-vET。
- -b,--number-nonblank：对非空输出行编号。
- -e：等价于-vE。
- -E,--show-ends：在每行结束处显示“$”。
- -n,--number：对输出的所有行编号。
- -s,--squeeze-blank：不输出多行空行。
- -t：与-vT 等价。
- -T,--show-tabs：将跳格字符显示为“^I”。
- -u：被忽略。
- -v,--show-nonprinting：使用“^”和“M-”引用，除 LFD 和 TAB 之外。
- --help：显示此帮助信息并退出。
- --version：输出版本信息并退出。

如果[文件]省略，或者[文件]为“-”，则读取标准输入。

应用实例如下。

（1）显示文本文件的内容。

cat 命令一个最简单的用处是显示文本文件的内容。例如，要查看 README 文本文件的内容，可以使用如下命令：

```
$ cat README
```

（2）合并文件。

有时需要将几个文件处理成一个文件，并将这种处理的结果保存到一个单独的输出文件。示例代码如下：

```
$cat README INSTALL Changlog > File1
```

把 README、INSTALL 和 Changlog 的文件内容加上行号（空白行不加）之后，将内容附

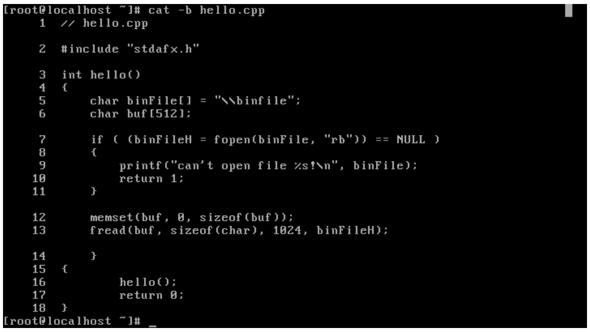
加到一个新文本文件 File1。

（3）先输出 f 文件的内容，然后输出标准输入的内容，最后输出 g 文件的内容。

```
#cat f – g
```

（4）对行进行编号。

这种功能对于程序文档的编制，以及法律和科学文档的编制很方便。打印在左边的行号使得参考文档的某一部分变得容易，这在编程、科学研究、业务报告，甚至立法工作中都是非常重要的。对行进行编号的功能有两个选项："-b"（对非空白行进行编号）和"-n"（对所有行进行编号），如图 2-1 所示。



```
[root@localhost ~]# cat -b hello.cpp
     1  // hello.cpp

     2  #include "stdafx.h"

     3  int hello()
     4  {
     5      char binFile[] = "\\binfile";
     6      char buf[512];

     7      if ( (binFileH = fopen(binFile, "rb")) == NULL )
     8      {
     9          printf("can't open file %s!\n", binFile);
    10          return 1;
    11      }

    12      memset(buf, 0, sizeof(buf));
    13      fread(buf, sizeof(char), 1024, binFileH);

    14      }
    15  {
    16          hello();
    17          return 0;
    18  }
[root@localhost ~]# _
```

图 2-1　使用 cat 命令给一个文本文件的所有行进行编号

（5）显示文件的非打印字符。

```
$ cat -t /etc/X11/XF86Config
...
# Multiple FontPath entries are allowed (they are concatenated together)
# By default, Red Hat 6.0 and later now use a font server independent of
# the X server to render fonts.
^IFontPath^I"/usr/X11R6/lib/X11/fonts/TrueType"
^IFontPath^I"unix/:7100"
EndSection
...
```

2.3 rm：删除文件

作用：删除指定的文件。

用法：rm [选项]...文件...

主要选项如下。

- -f,--force：强制删除。忽略不存在的文件，不提示确认。
- -i：在删除前需要确认。

– -I：在删除超过 3 个文件或者递归删除前要求确认。此选项比-i 提示内容更少，但同样可以阻止大多数错误发生。
– --interactive[=WHEN]：根据指定的 WHEN 进行确认提示，如 never、once(-I)或者always(-i)。如果此参数不加 WHEN，则总是提示。
– --one-file-system：递归删除一个层级时，跳过所有不符合命令行参数的文件系统上的文件。
– --no-preserve-roo：不特殊对待“/”。
– --preserve-root：不允许删除“/”（默认）。
– -r,-R,--recursive：递归删除目录及其内容。
– -v,--verbose：详细显示进行的步骤。
– --help：显示此帮助信息并退出。
– --version：显示版本信息并退出。

在默认情况下，rm 不会删除目录。使用—recursive（-r 或-R）选项可删除每个给定的目录，以及其下所有的内容。

应用实例如下。

（1）删除系统安装文件 install.log。

```
#rm install.log
```

使用 rm 命令要小心，因为一旦文件被删除，恢复比较困难。为了防止这种情况的发生，可以使用 i 选项来逐个确认要删除的文件。如果用户输入 y，文件将被删除；如果输入任何其他字符，文件不会被删除。

（2）删除一个目录 test。

```
#rm –rf test
```

如果任何一级的目录有文件，要用 rm -rf 命令强制删除。不过使用 rm -rf 命令删除时有点不安全，还是用 rf -ri 比较好，因为在删除的时候会有警告提示，这对于安全操作来说是很重要的。

2.4 less：分屏显示文件

作用：less 命令的功能几乎和 more 命令一样，也用来按页显示文件，不同之处在于 less 命令在显示文件时，用户既可以向前又可以向后翻阅文件。

用法：less[选项]文件

主要选项如下。

– -c：从顶部（从上到下）刷新屏幕，并显示文件内容，而不是通过底部滚动完成刷新。
– -f：强制打开文件，二进制文件显示时，不提示警告。
– -i：搜索时忽略大小写，除非搜索串中包含大写字母。

- -I：搜索时忽略大小写，除非搜索串中包含小写字母。
- -m：显示读取文件的百分比。
- -M：显示读取文件的百分比、行号及总行数。
- -N：在每行前输出行号。
- -p pattern：搜索 pattern，如在/etc/profile 下搜索单词 MAIL，就用 less -p MAIL /etc/profile。
- -s：把连续多个空白行作为一个空白行显示。
- -Q：在终端下不响铃。

命令内部操作介绍如下。

按键功能如下。

- b：向后翻一页。
- d：向后翻半页。
- h：显示帮助界面。
- Q：退出 less：命令。
- u：向前滚动半页。
- y：向前滚动一行。
- 空格键：滚动一行。
- Enter 键：滚动一页。

1. 向前搜索

- /：使用一个模式进行搜索，并定位到下一个匹配的文本。
- n：向前查找下一个匹配的文本。
- N：向后查找前一个匹配的文本。

2. 向后搜索

- ?：使用模式进行搜索，并定位到前一个匹配的文本。
- n：向后查找下一个匹配的文本。
- N：向前查找前一个匹配的文本。

3. 全屏导航

- Ctrl+F：向前移动一屏。
- Ctrl+B：向后移动一屏。
- Ctrl+D：向前移动半屏。
- Ctrl+U：向后移动半屏。

4. 单行导航

- j：向前移动一行。
- k：向后移动一行。

5. 其他导航

- G：移动到最后一行。
- g：移动到第一行。

– q/ZZ：退出 less 命令。

6．编辑文件

v：进入编辑模式，使用配置的编辑器编辑当前文件。

7．标记导航

当使用 less 命令查看大文件时，可以在任何一个位置作标记，可以通过命令导航到标有特定标记的文本位置。

– ma：使用 a 标记文本的当前位置。
– 'a：导航到标记 a 处。

应用实例如下。

（1）查看系统安装文件 install.log 的百分比、行号及总行数。

```
#less -M install.log
```

统计信息显示在底部，如图 2-2 所示。

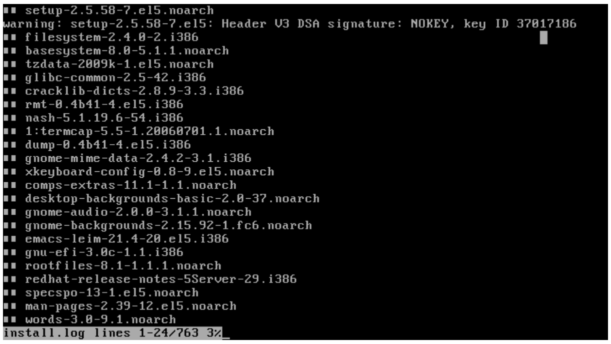


图 2-2　查看系统安装文件 install.log 的百分比、行号及总行数

（2）查看 hello.cpp 文件并且搜索一个字符 char。

```
#less –p char hello.cpp
```

结果如图 2-3 所示。此时可以使用 vi 编辑器对该文件进行编辑。

图 2-3　查看 hello.cpp 文件并且搜索一个字符 char

2.5 cp：复制文件

作用：文件或目录的复制。

用法：cp [选项]... [-T] 源文件 目标文件

　　　cp [选项]... 源文件... 目录

　　　cp [选项]... -t 目录 源文件...

将<源文件>文件复制至<目标文件>，或将多个<源>复制至<目录>。

主要选项如下。

- -a：等同于-dpR 参数一起使用。在复制过程中尽可能保留源文件或目录的属性。
- -b：给被覆盖的文件进行备份。例如，将 file1 复制到 file2 时，如果 file2 存在，则把 file2 覆盖掉，用这个参数就能同时生成一个名为 file2～的文件，注意文件名后有“～”作为结尾。
- -d：不间接引用符号链接，保持源文件和目标文件之间的硬链接关系。
- -f：删除已存在的目标文件。
- -i：覆盖已存在的目标文件之前给出警告提示。这是人机交互的选项。
- -l：建立文件硬链接，而不是复制。
- -p：保存源文件或目录的最后修改时间和模式，如要保持目标目录或文件的属主、组、权限和最后访问（或修改）时间。
- -R：复制目录时，包括此目录下所有的子目录和文件。
- -s：建立源文件的符号链接，而不是复制源文件。源文件名必须用绝对路径。
- -r：复制目录时，包括此目录下所有的子目录和文件；-r 选项不同于-R 之处在于尝试打开目的地文件前先删除已存在的目的地文件。
- -i：人机交互模式，显示覆盖已存在的文件或目录的警告信息。
- -v：在复制之前，输出每个文件的名字。
- -Z,--context[=CTX]：设定 SELinux 安全上下文为默认类型或指定为 CTX。如果指定
- 源文件：指定源文件列表。在默认情况下，cp 命令不能复制目录；如果要复制目录，则必须使用-R 选项。
- 目标文件：指定目标文件。当“源文件”为多个文件时，要求“目标文件”为指定的目录。

应用实例如下。

（1）参数-i 的应用示例。

```
# cp  -i Kernel.rpm Plist
cp：是否覆盖"Plist"?y
```

上述代码是人机交互参数-i 的运用。在将源文件复制到目标文件时，如果目标文件存在，则提出警告。

（2）参数-a 的应用示例及说明。

把源目录复制到另外的一个目标目录，相当于克隆了一个目录，源目录和目标目录所在用户组、权限及修改或访问时间保持一致。

```
# cp -a mydir youdir
```

复制目录 mydir 到 youdir，如果 youdir 目录不存在，则创建，并且把 mydir 中所有的文件下级子目录都复制到 youdir 目录中。也就是说 mydir 目录的文件和 youdir 是一样的。-a 参数等同于-dpR，并且源目录和目标目录的属性（用户组、权限、最后访问或修改时间）一致。

（3）参数-b 的应用示例。

把源文件复制为目标文件，并且尝试-b 参数的运用。

cp fonts.scale fonts.dir 把一个文件复制为另一个文件。

```
# cp -bi fonts.scale fonts.dir
cp：是否覆盖"fonts.dir"? y
```

> **注意**
> 因为用了-i 参数，如果目标文件存在，则报警提示。

2.6 mv：更改文件名

作用：mv 可以将源文件重命名为目标文件，或将源文件移动至指定目录。其实 mv 这个过程也可以理解为改名过程，所以说 mv 工具能修改文件名和目录名。

用法：mv [选项]... [-T]源目的

mv [选项]... 源...目录

mv [选项]... -t 目录源...

主要选项如下。

长选项必须使用的参数对于短选项也是必须使用的。

- --backup[=CONTROL]：为每个已存在的目标文件创建备份。
- -b：类似--backup 但不接受参数。
- -f,--force：覆盖前不询问。
- -i,--interactive：覆盖前询问。
- -n,--no-clobber：不覆盖已存在文件。

如果指定了-i、-f、-n 中的多个，仅最后一个生效。

- --strip-trailing-slashes：去掉每个源文件参数尾部的斜线。
- -S,--suffix=SUFFIX：替换常用的备份文件后缀。
- -t,--target-directory=DIRECTORY：将所有参数指定的源文件或目录移动至指定目录。

– -T,--no-target-directory：将目标文件视作普通文件处理。
– -u,--update：只在源文件文件比目标文件新，或目标文件不存在时才进行移动。
– -v,--verbose：详细显示进行的步骤。
– --help：显示此帮助信息并退出。
– --version：显示版本信息并退出。

备份文件的后缀为"~"，除非以--suffix 选项或是 SIMPLE_BACKUP_SUFFIX 环境变量指定。版本控制的方式可通过--backup 选项或 VERSION_CONTROL 环境变量来选择。以下是可用的变量值。

Ø none,off：不进行备份（即使使用了--backup 选项）。
Ø numbered,t：备份文件加上数字进行排序。
Ø existing,nil：若有数字的备份文件已经存在则使用数字，否则使用普通方式备份。
Ø simple,never：永远使用普通方式备份。
Ø -Z, --context：设定目的地 SELinux 安全上下文文件为默认类型。

应用实例如下。

（1）移动 crodw.sh 到 23.sh，其实就是把文件名改为 123.sh。

```
mv crodw.sh 23.sh
```

（2）mv 修改目录。

mv 用于把文件（或目录）从一个地方移到另一个地方。如果另一个地方没有和源文件（或目录）同名的文件（或目录），就创建。这也相当于改名。

```
# mv kernel/ kernelBak1
```

2.7 grep：查找字符串

作用：查找文件中符合条件的字符串。grep（global search regular expression(RE) and print out the line，全面搜索正则表达式并把行打印出来）是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹配的行打印出来。UNIX 的 grep 家族包括 grep、egrep 和 fgrep。egrep 和 fgrep 的命令只跟 grep 略有不同。egrep 是 grep 的扩展，支持更多的 re 元字符，fgrep 就是 fixed grep 或 fast grep，它们把所有的字母都看作单词，也就是说，正则表达式中的元字符表示其自身的字面意义，不再特殊。Linux 使用 GNU 版本的 grep。它功能更强，可以通过-G、-E、-F 命令行选项来使用 egrep 和 fgrep 的功能。grep 的工作方式是这样的：它在一个或多个文件中搜索字符串模板。如果模板包括空格，则必须被引用，模板后的所有字符串被看作文件名。搜索的结果被送到屏幕，不影响原文件内容。grep 可用于 Shell 脚本，因为 grep 通过返回一个状态值来说明搜索的状态。如果模板搜索成功，则返回 0；如果搜索不成功，则返回 1；如果搜索的文件不存在，则返回 2。我们利用这些返回值就可进行一些自动化的文本处理工作。

用法：grep [选项] <pattern> <files>

1．匹配模式选择

- -E,--extended-regexp：扩展正则表达式 egrep。
- -F,--fixed-strings：一个换行符分隔的字符串的集合 fgrep。
- -G,--basic-regexp：基本正则。
- -P,--perl-regexp：调用的 perl 正则。
- -e,--regexp=PATTERN：后面跟正则模式，默认无。
- -f,--file=FILE：从文件中获得匹配模式。
- -i,--ignore-case：不区分大小写。
- -w,--word-regexp：匹配整个单词。
- -x,--line-regexp：匹配整行。
- -z,--null-data：一个 0 字节的数据行，但不是空行。

2．杂项

- -s,--no-messages：不显示错误信息。
- -v,--invert-match：显示不匹配的行。
- -V,--version：显示版本号。
- --help：显示帮助信息。

3．输入控制

- -m,--max-count=NUM：匹配的最大数。
- -b,--byte-offset：打印该行所在的块号码。
- -n,--line-number：显示加上匹配所在的行号。
- --line-buffered：刷新输出每一行。
- -H,--with-filename：当搜索多个文件时，显示匹配文件名前缀。
- -h,--no-filename：输出时不显示文件名前缀。
- --label=LABEL：将 LABEL 作为标准输入文件名前缀。
- -o,--only-matching：只显示一行中匹配 PATTERN 的部分。
- -q,--quiet,--silent：不显示任何信息。
- --binary-files=TYPE：假定二进制文件的 TYPE 类型。TYPE 可以是'binary'、'text'或'without-match'。
- -a,--text：匹配二进制的信息。
- -I：不匹配二进制的信息。
- -d,--directories=ACTION：目录操作，如读取、递归、跳过。
- -D,--devices=ACTION：设置对设备、FIFO、管道的操作，如读取、跳过。
- -R,-r,--recursive：递归调用。
- --include=PATTERN：只查找匹配 FILE_PATTERN 的文件。
- --exclude=PATTERN：跳过匹配 FILE_PATTERN 的文件和目录。
- --exclude-from=FILE：跳过所有除 FILE 以外的文件。
- -L,--files-without-match：匹配多个文件时，显示不匹配的文件名。

– -l,--files-with-matches：匹配多个文件时，显示匹配的文件名。
– -c,--count：显示匹配了多少次。
– -Z,--null：在 FILE 文件最后打印空字符。

4．文件控制

– -B,--before-context=NUM：打印匹配本身及前面的几行，由 NUM 控制。
– -A,--after-context=NUM：打印匹配本身及随后的几行，由 NUM 控制。
– -C,--context=NUM：打印匹配本身及随后、前面的几行，由 NUM 控制。
– -NUM：与-C 的用法相同。
– --color[=WHEN]：使用标志高亮匹配字串。
– -u,--unix-byte-offsets：当 CR 字符不存在时，报告字节偏移（MSDOS 模式）。

应用实例如下。

（1）查找文件“file.php”中是否包含字符串“html”。

```
# grep 'html' file.php
```

（2）检查/etc/passwd 文件中是否有可疑用户。

Linux 中的/etc/passwd 文件是存储系统用户密码等重要信息的文件，黑客入侵系统后往往会使用在 passwd 文件中增加特权用户的方法为自己留个后门。所以，我们要经常查看该文件。如果系统用户较少，可以采用直接查看 passwd 文件的方法，命令为“cat /etc/passwd”。如果系统有成百上千个用户，那么直接查看就不行了，不过只要检测其中是否有 UID 为 0 的特权用户就行了，这时可以使用以下命令来实现。

```
# grep '0:0' /etc/passwd
```

说明

grep 命令用于查找内容包含指定的范本样式的文件，如果发现某个文件的内容符合所指定的范本样式，预设 grep 命令会把含有范本样式的那一列显示出来。若不指定任何文件名称，或是所给予的文件名为“-”，则 grep 命令会从标准输入设备读取数据。要用好 grep 这个工具，其实就是要写好正则表达式。下面是正则表达式的应用实例。

（3）通过管道过滤 ls -l 输出的内容，只显示以 a 开头的行。

```
ls -l | grep '^a'
```

（4）显示所有以 d 开头的文件中包含 test 的行。

```
$ grep 'test' d*
```

（5）显示在 aa、bb、cc 文件中匹配 test 的行。

```
$ grep 'test' aa bb cc
```

（6）显示所有包含每个字符串至少有 5 个连续小写字符的字符串的行。

```
$ grep ' [a-z] ' aa
```

2.8 head：显示文件头部

作用：显示一个文件内容的前多少行。

用法：head [选项] [文件]

将每个指定文件的头 10 行显示到标准输出。

如果指定了多于一个文件，则在每一段输出前会给出文件名作为文件头。

如果不指定文件，或者文件为“-”，则从标准输入读取数据。

主要选项如下。

长选项必须使用的参数对于短选项也是必须使用的。

- -c,--bytes=[-]K：显示每个文件的前 *K* 字节内容。如果附加“-”参数，则除了每个文件的最后 *K* 字节数据外，显示剩余全部内容。
- -n,--lines=[-]K：显示每个文件的前 *K* 行内容。如果附加“-”参数，则除了每个文件的最后 *K* 行外，显示剩余全部内容。
- -q,--quiet,--silent：不显示包含给定文件名的文件头。
- -v,--verbose：总是显示包含给定文件名的文件头。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

应用实例如下。

（1）显示/etc/profile 的前 10 行内容。

```
# head -n 10 /etc/profile
```

（2）将/etc/named.conf 中前 3 行的内容发送至标准输出。

```
$ head -n 3 /etc/ named.conf
/etc/named.conf <==
// generated by named－bootconf.pl
options {
```

> **说明**
>
> 命令“head file1 file2 file3”和命令“cat file1 file2 file3 | head”有重大差别。前者将打印每个文件指定行数的内容，不同文件的内容之间用头信息隔开，头信息以“==>”后跟文件名开头。后者将打印由 cat 命令后所列文件组成的输入流中指定行数的内容，但将把输入流作为单个文件对待。

2.9 tail：显示文件尾部

作用：显示一个文件内容的最后多少行。

用法：tail[选项] [文件]

主要选项如下。

- --retry：即使 tail 开始时就不能访问或者在 tail 运行后不能访问，也仍然不停地尝试打开文件。--只与-f 合用时有效。
- -c,--bytes=N：输出最后 *N* 字节。
- -f,--follow[={name|descriptor}]：当文件增长时，输出后续添加的数据；-f,--follow 及 --follow=descriptor 含义相同。
- -n,--lines=N：输出最后 *N* 行，而非默认的最后 10 行。
- --pid=PID：与-f 合用，表示在进程 ID、PID 死掉之后结束。
- -q,--quiet,--silent：从不输出给出文件名的首部。
- -s,--sleep-interval=S：与-f 合用，表示在每次反复的间隔休眠 *S* 秒。
- -v,--verbose：总是输出给出文件名的首部。
- --help：显示帮助信息后退出。
- --version：输出版本信息后退出。

如果 N（字节或者行数）的第一个字符为`+'，那么从每个文件的开头算起的第 *N* 项开始显示，否则，显示该文件的最后 *N* 项。*N* 可以有一个倍数前缀：b 表示 512，k 表示 1024，m 表示 1048576（1 兆）。第一个选项-VALUE 或+VALUE 以-nVALUE 或-n+VALUE 方式看待，除非 VALUE 含有[bkm]后缀倍数中的一个，在这种情况下，它被看作-cVALUE 或者-c+VALUE。

使用--follow(-f)时，tail 默认后接文件描述符，这意味着即使 tail 显示的文件改名了，tail 仍然可以追踪到其末尾部分。如果确实希望查询文件的实际名而非文件描述符（例如，日志轮转时），这种默认的操作就不是用户所期望的了。在这种情况下应使用--follow=name。这将导致 tail 通过周期性地重新打开所指定的文件来追踪其是否被删除了，或者被其他程序重新创建了。

应用实例如下。

（1）显示/etc/profile 的最后 5 行内容。

```
#tail -n 5 /etc/profile
```

（2）将/etc/named.conf 中后 3 行的内容发送至标准输出并显示头信息。

```
$tail -n 3 -v /etc/named.conf
==> /etc/named.conf <==
};
include "/etc/rndc.key";
```

> **说明**
>
> 如果 tail 正在读取的文件不止一个，那么各行内容之间将用标准头信息隔开，以指明它们来自哪个文件，标准头信息以“==>”开头，该选项用于监视系统日志非常合适。例如，在单独的终端窗口中执行的“tail -f /var/log/access.log”将持续打印每次单击后新添加的 Apache 访问日志条目，一直到按 Ctrl+C 组合键停止它为止。

2.10 sort：按顺序显示文件内容

作用：按顺序显示文件内容。

用法：sort [选项]...[文件]...

排序选项如下。

- -b,--ignore-leading-blanks：忽略前导的空白区域。
- -d,--dictionary-order：只考虑空白区域和字母字符。
- -f,--ignore-case：忽略字母大小写。
- -g,--general-numeric-sort：按照常规数值排序。
- -i,--ignore-nonprinting：只排序可打印字符。
- -M,--month-sort：比较（未知）<"一月"<...<"十二月"，在<C–ALL=C 时为（unknown）<'JAN'<...'DEC'。
- -h,--human-numeric-sort：使用易读性数字（例如，2K、1G）。
- -n,--numeric-sort：根据字符串数值比较。
- -R,--random-sort 根据随机 Hash 排序。
- --random-source=文件：从指定文件中获得随机字节。
- -r,--reverse：逆序输出排序结果。
- --sort=WORD：按照 WORD 指定的格式排序。一般数字-g，高可读性-h，月份-M，数字-n，随机-R，版本-V。
- -V,--version-sort：在文本内进行自然版本排序。

其他选项如下。

- --batch-size=NMERGE：一次最多合并 NMERGE 个输入；如果输入更多则使用临时文件。
- -c,--check,--check=diagnose-first：检查输入是否已排序，若已有序则不进行操作。
- -C,--check=quiet,--check=silent：类似-c，但不报告第一个无序行。
- --compress-program=程序：使用指定程序压缩临时文件；使用该程序的-d 参数解压缩文件。
- --debug：为用于排序的行添加注释，并将可能有问题的用法输出到标准错误输出。
- --files0-from=文件：从指定文件读取以 NUL 终止的名称，如果该文件被指定为“-”，则从标准输入读文件名。
- -k,--key=位置 1[,位置 2]：在位置 1 开始一个 key，在位置 2 终止（默认为行尾）。

- -m,--merge：合并已排序的文件，不再进行排序。
- -o,--output=文件：将结果写入到文件而非标准输出。
- -s,--stable：禁用 last-resort 算法。
- -S,--buffer-size=大小：指定主内存缓存大小。
- -t,--field-separator=分隔符：使用指定的分隔符代替非空格到空格的转换。
- -T,--temporary-directory=目录：使用指定目录而非$TMPDIR 或/tmp 作为临时目录，可用多个选项指定多个目录。
- --parallel=N：将同时运行的排序数改变为 N。
- -u,--unique：配合-c，严格校验排序；不配合-c，则只输出一次排序结果。
- -z,--zero-terminated：以 0 字节而非新行作为行尾标志。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

说明

sort 命令有许多非常实用的选项，这些选项最初是用来对数据库用法的文件内容进行各种排序操作的。实际上，可以认为 sort 命令是一个非常强大的数据管理工具，用来管理内容类似数据库记录的文件。sort 命令将逐行对文件中的内容进行排序，如果两行的首字符相同，该命令将继续比较这两行的下一个字符，如果还相同，将继续进行比较。如果不指定输入文件，使用“–”，则表示排序内容来自标准输入。sort 排序是根据从输入行抽取的一个或多个关键字进行比较来完成的。排序关键字定义了用来排序的最小的字符序列。默认情况下以整行为关键字按 ASCII 字符顺序进行排序。指定的大小可以使用以下单位之一：内存使用率%——1%、b——1、K——1024（默认），M、G、T、P、E、Z、Y 等，依此类推。

应用实例如下。

（1）对密码文件进行排序。

```
# sort /etc/passwd > /etc/passwd－new
# mv /etc/passwd－new /etc/passwd
```

注意

不能将输出直接发送到输入文件，因为这会破坏输入文件。这就是为何需要将它发送到临时文件中，然后将该文件重命名为/etc/passwd 的原因。如果想倒转排序的次序，则应当使用–r 选项。还可以用–u 选项来禁止打印相同的行。

（2）用“sort -m”排序合并多个日志。

许多大型的 Web 服务使用 DNS 轮循来实现负载均衡。对于使用多个同样角色的服务器做

前台的 Web 服务，多个服务器的分布使得日志的分析统计也变得有些麻烦。可以使用 sort 合并，示例代码如下：

```
sort -m -t " " -k 4 -o log_all log1 log2 log3
```

（3）sort 的-k 选项的使用。

从公司英文名称的第二个字母开始进行排序。

```
$ sort -t ' ' -k 1.2 facebook.txt
baidu 100 5000
sohu 100 4500
google 110 5000
guge 50 3000
```

使用了-k 1.2，表示对从第一个域的第二个字符开始到本域的最后一个字符为止的字符串进行排序。读者会发现 baidu 因为第二个字母是 a 而名列榜首；sohu 和 google 第二个字符都是 o，但 sohu 的 h 在 google 的 o 前面，所以二者分别排在第二和第三；guge 只能屈居第四了。只针对公司英文名称的第二个字母进行排序，如果相同则按照员工工资进行降序排序。

```
$ sort -t ' ' -k 1.2,1.2 -nrk 3,3 facebook.txt
 baidu 100 5000
 google 110 5000
sohu 100 4500
 guge 50 3000
```

由于只对第二个字母进行排序，所以使用了-k 1.2,1.2 的表示方式，表示我们“只”对第二个字母进行排序（为什么使用-k 1.2 不行？当然不行，因为省略了 End 部分，这就意味着将对从第二个字母起到本域最后一个字符为止的字符串进行排序）。对员工工资进行排序，也使用了-k 3,3，这是最准确的表述，表示“只”对本域进行排序，因为如果省略了后面的 3，就变成了“对从第三个域开始到最后一个域位置的内容进行排序”了。

2.11 uniq：忽略文件中的重复行

作用：文件经过处理后，在它的输出文件中可能会出现重复的行。例如，使用 cat 命令将两个文件合并后，再使用 sort 命令进行排序，就可能出现重复行。这时可以使用 uniq 命令将这些重复行从输出文件中删除，只留下每条记录的唯一样本。

用法：uniq [选项]...[输入[输出]]

主要选项如下。

长选项必须使用的参数对于短选项也是必须使用的。

- -c,--count：在每行前加上表示相应行目出现次数的前缀编号。

- -d,--repeated：只输出重复的行。
- -D,--all-repeated[=delimit-method：显示所有重复的行。
- delimit-method={none(default),prepend,separate}：以空行为界限。
- -f,--skip-fields=N：在比较时跳过前 *N* 列。
- -i,--ignore-case：在比较的时候不区分大小写。
- -s,--skip-chars=N：在比较时跳过前 *N* 个字符。
- -u,--unique：只显示唯一的行。
- -z,--zero-terminated：使用'\0'作为行结束符，而不是新换行。
- -w,--check-chars=N：对每行第 *N* 个字符以后的内容不作对照。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

应用实例如下。

首先看一下示例文件。

```
$ cat happy
Happy Birthday to You!
Happy Birthday to You!
Happy Birthday Dear cjh!
Happy Birthday to You!
```

（1）显示文件 happy 中不重复的行。

```
 # uniq -u happy
Happy Birthday Dear cjh!
Happy Birthday to You!
```

（2）显示文件 happy 中不重复的行，从第二个字段的第二个字符开始进行比较。

```
#uniq -u -1 +1 happy
```

（3）用-c 选项从 uniq 中获取一些统计信息。

```
#sort happy | uniq -dc
3 Happy Birthday to You!
```

这里 uniq 命令借助管道命令从标准输入设备读取文件，首先使用 sort 命令对文件进行排序，然后对排序结果进行操作。

（4）删除文件中重复的行并输出到一个新文件。

```
#sort happy|uniq-uniqhappy
```

操作过程如图 2-4 所示。

注意

请不要使用 uniq 或任何其他工具从包含财务或其他重要数据的文件中除去重复行。在这种情况下，重复行几乎总是表示同一金额的另一个交易，将它除去会给会计部造成许多困难。

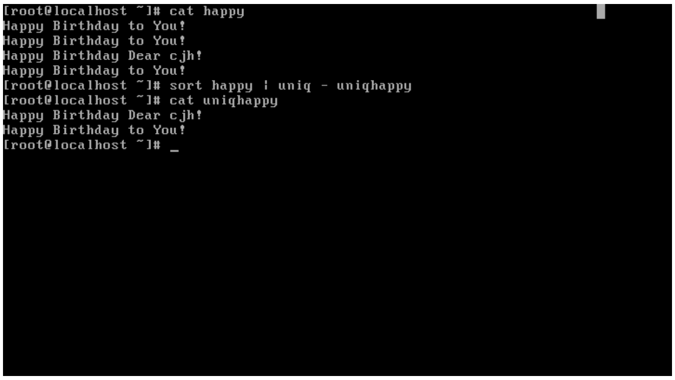


图 2-4　删除文件中重复的行并且输出到一个新文件

（5）高级应用。

就算 uniq 对完整的行进行比较，它仍然会很有用，但是那并非该命令的全部功能。特别方便的是，使用-f 选项，后面跟着要跳过的字段数，它能够跳过给定数目的字段。在查看系统日志时非常有用。通常，某些项要被复制许多次，这使得查看日志很困难。使用简单的 uniq 无法完成任务，因为每一项都以不同的时间戳记开头。但是如果告诉它跳过所有的时间字段，日志一下子就会变得便于管理。试一试 uniq -f 3 /var/log/messages。还有另一个选项-s，它的功能就像-f 一样，但是跳过给定数目的字符。可以一起使用-f 和-s，uniq 先跳过字段，再跳过字符。如果只想使用一些预先设置的字符进行比较，试试-w 选项。

2.12　diff：比较两个文件

作用：找出两个文件的不同点。

用法：diff [选项] 源文件 目标文件

主要选项如下。

下面是 GNU 所接受的 diff 所有选项的概要。大多数选项有两个相同的名字，一个是单个地跟在“-”后面的字母，另一个是由“--”引出的长名字。多个单字母选项（除非它们产生歧义）能够组合为单行的命令行语法，-ac 等同于 -a –c。长命名的选项能被缩短到它们的名字的任何唯一的前缀。用 [和] 括起来显示产生歧义的选项。

– -行数（一个整数）：显示上下文行数（一个整数）。这个选项自身没有指定输出格式，这是没有效果的，除非和-c 或者-u 组合使用。这是已废置的选项，对于正确的操作，上

下文至少要有两行。
- -a：所有的文件都视为文本文件来逐行比较，甚至它们似乎不是文本文件。
- -b：忽略空格引起的变化。
- -B：忽略插入、删除空行引起的变化。
- --brief：仅报告文件是否相异，在乎差别的细节。
- -c：使用上下文输出格式。
- -C：行数（一个整数）。
- --context[=lines]：使用上下文输出格式，显示以指定行数（一个整数），或者是三行（当行数没有给出时）。对于正确的操作，上下文至少要有两行。
- --changed-group-format=format：使用 format 输出一组包含两个文件的不同处的行，其格式是 if-then-else。
- -d：改变算法也许发现变化的一个更小的集合，这个选项会使 diff 命令执行速度变慢。
- -D name：合并 if-then-else 格式输出，预处理宏（由 name 参数提供）条件。
- -e,--ed：输出为一个有效的 ed 脚本。
- --exclude=pattern：比较目录的时候，忽略和目录中与 pattern（样式）相匹配的。
- --exclude-from=file：比较目录的时候，忽略和目录中与任何包含在 file（文件）中的样式相匹配的文件和目录。
- --expand-tabs：在输出时扩展 tab 为空格，保护输入文件的 tab 对齐方式。
- -f：产生一个类似 ed 脚本的输出，但是它们在文件中出现的顺序有所改变。
- -F regexp：在上下文和统一格式中，对于每一大块的不同，显示出匹配 regexp.的一些前面的行。
- --forward -ed：产生类似 ed 脚本的输出，但是它们在文件中出现的顺序有所改变。
- -H：使用启发规则加速操作那些有许多离散的小差异的大文件。
- --horizon-lines=lines：比较给定行数的有共同前缀的最后行，以及有共同后缀的最前行。
- -i：忽略大小写。
- -I regexp：忽略由插入或删除行（由 regexp 参数提供参考）带来的改变。
- --ifdef=name：合并 if-then-else 格式输出，预处理宏（由 name 参数提供）条件。
- --ignore-all-space：在比较行的时候忽略空白。
- --ignore-blank-lines：忽略插入和删除空行。
- --ignore-case：忽略大小写。
- --ignore-matching-lines=regexp：忽略插入、删除行。
- --ignore-space-change：忽略空白的数量。
- --initial-tab：在文本行（无论是常规的还是格式化的前后文关系）前输出 tab 代替空格。引起的原因是 tab 对齐方式看上去像是常规的一样。
- -l：产生通过 pr 编码的输出。
- -L label--label=label：使用 label 给出的字符在文件头代替文件名输出。
- --left-column：以并列方式印出两个公共行的左边。
- --line-format=format：使用 format 格式输出所有的行。

- --minimal：改变算法也许发现变化的一个更小的集合。这会使 diff 变慢（有时更慢）。
- -n：输出 RC-格式 diffs；除了每条指令指定的行数受影响外，像-f 一样。
- -N：在比较目录时，若文件 A 仅出现在某个目录中，预设会显示 Only in 目录；文件 A 若使用-N 参数，则 diff 会将文件 A 与一个空白的文件比较。
- --new-file：在目录比较中，如果哪个文件只在其中的一个目录中找到，那么这个文件被视为在另一个目录中是一个空文件。
- --new-group-format=format：使用 format 以 if-then-else 格式输出只在第二个文件中取出的一个行组。
- --new-line-format=format：使用 format 以 if-then-else 格式输出只在第二个文件中取出的一行。
- --old-group-format=format：使用 format 以 if-then-else 格式输出只在第一个文件中取出的一个行组。
- --old-line-format=format：使用 format 以 if-then-else 格式输出只在第一个文件中取出的一行。
- -p：显示带有 c 函数的改变。
- -P：在目录比较中，如果哪个文件只在其中的一个目录中找到，那么它被视为在另一个目录中是一个空文件。
- --paginate：产生通过 pr 编码的输出。
- -q：仅报告文件是否相异，不报告详细的差异。
- -r：当比较目录时，递归比较任何找到的子目录。
- --rcs：输出 RC-格式 diffs；除了每条指令指定的行数受影响外，像-f 一样。
- --recursive：当比较目录时，递归比较任何找到的子目录。
- --report-identical-files：若没有发现任何差异，仍然显示信息。
- -s：报告两个文件相同。
- -S file：当比较目录时，由 file 开始。这用于继续中断了的比较。
- --sdiff-merge-assist：打印附加的信息去帮助 sdiff.sdiff 在运行 diff 时使用这些选项。这些选项不是特意为使用者直接使用而准备的。
- --show-c-function：显示带有 c 函数的改变。
- --show-function-line=regexp：在上下文和统一的格式，对于每一大块的差别，显示出匹配 regexp.的一些前面的行。
- --side-by-side：使用并列的输出格式。
- --speed-large-files：使用启发规则加速操作那些有许多离散的小差异的大文件。
- --starting-file=file：当比较目录时，由 file 开始。这用于继续中断了的比较。
- --suppress-common-lines：在并列格式中不打印出公共行。
- -t：在输出时扩展 tab 为空格，保护输入文件的 tab 对齐方式。
- -T：在文本行（无论是常规的还是格式化的前后文关系）前输出 tab 代替空格引起的原因是 tab 对齐方式看上去像是常规的一样。
- --text：所有的文件都视为文本文件来逐行比较，甚至它们似乎不是文本文件。

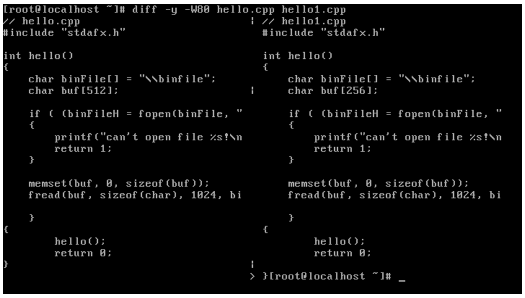
- -u：使用统一的输出格式。
- --unchanged-group-format=format：使用 format 输出两个文件的公共行组，其格式是 if-then-else。
- --unchanged-line-format=format：使用 format 输出两个文件的公共行，其格式是 if-then-else。
- --unidirectional-new-file：在目录比较中，如果哪个文件只在其中的一个目录中找到，那么它被视为在另一个目录中是一个空文件。
- -U lines：以合并的方式来显示文件内容的不同。
- --unified[=lines]：使用前后关系格式输出，显示指定行数（一个整数）或者三行（当行数没有给出时）。对于正确的操作，上下文至少要有两行。
- --version：输出 diff 版本号。
- -w：在比较行时忽略空格。
- -W columns：指定栏宽。
- --width=columns：在并列格式输出时，使用指定的列宽。
- -x pattern：比较目录的时候，忽略和目录中与 pattern（样式）相匹配的。
- -X file：比较目录的时候，忽略和目录中与任何包含在 file（文件）中的样式相匹配的文件和目录。
- -y：使用并列格式输出。

说明

diff 以逐行的方式比较文本文件的异同处。如果源文件和目标文件都是目录，则 diff 比较两个目录中相应的文件，按照字母次序排序。diff 的输出用法有 3 种，包括列举模式、命令模式和上下文模式，其中命令模式又分为两种，即 ed 命令模式和 RCS（Revision Control System，版本控制系统）命令模式。

应用实例如下。

使用并排（side-by-side）用法查看文件差异，如图 2-5 所示。



```
[root@localhost ~]# diff -y -W80 hello.cpp hello1.cpp
// hello.cpp                                  | // hello1.cpp
#include "stdafx.h"                             #include "stdafx.h"

int hello()                                     int hello()
{                                               {
    char binFile[] = "\\binfile";                   char binFile[] = "\\binfile";
    char buf[512];                            |     char buf[256];

    if ( (binFileH = fopen(binFile, "             if ( (binFileH = fopen(binFile, "
    {                                               {
        printf("can't open file %s!\n                 printf("can't open file %s!\n
        return 1;                                       return 1;
    }                                               }

    memset(buf, 0, sizeof(buf));                    memset(buf, 0, sizeof(buf));
    fread(buf, sizeof(char), 1024, bi               fread(buf, sizeof(char), 1024, bi

    }                                               }
{                                               {
        hello();                                        hello();
        return 0;                                       return 0;
}                                             |
                                              > }[root@localhost ~]# _
```

图 2-5　并排（side-by-side）用法输出

这是两个相似却又有所不同的文件：hello.cpp 和 hello1.cpp，使用并排（side-by-side）用法可以直观地比较源代码文件的差异。尽管出现了截行，还是可以非常清楚地显示两个文件的区

别。-W 选项指定输出列的宽度。diff 接受这个宽度，并且分配给两个文件各 40 行。字符“>”表示该行在目的文件中，字符“<”表示该行在源文件中，字符“|”标记出两个文件中不相同的行。

2.13 diffstat：diff 结果的统计信息

作用：读取 diff 的输出结果，然后统计各文件的插入、删除、修改等差异计量。

用法：diffstat[选项] [文件]

主要选项如下。

- -n<文件名长度>：指定文件名长度，指定的长度必须大于或等于所有文件中最长的文件名。
- -p<文件名长度>：与-n 参数相同，但此处的<文件名长度>包括文件的路径。
- -w：指定输出时栏位的宽度。
- -V：显示版本信息。

应用实例如下。

比较两个文件，代码如下：

```
#diffstat 1.txt 2.txt
 0 files changed
```

2.14 file：测试文件内容

作用：通过探测文件内容来判断文件类型。

用法：file [选项]文件名

主要选项如下。

- -b：不输出文件名（简要模式）。
- -c：检查时打印输出幻数文件的解析结果。常与-m 一起使用，用来在安装幻数文件之前调试它。
- -f 命名文件：从在参数表前的命名文件中读出将要检查的文件名（每行一个文件）。要有命名文件，或者至少有一个文件名参数；如果要检查标准输入，使用``-作为文件参数。
- -m list：指定包含幻数的文件列表。可以是单个文件，也可以是用冒号分隔的多个文件。
- -n：每检查完一个文件就强制刷新标准输出。仅在检查一组文件时才有效。一般在将文件类型输出到管道时才使用此选项。
- -v：打印程序版本并退出。
- -z：试图查看压缩文件内部信息。
- -L：直接显示符号链接所指向的文件的类型。

– -s：使用-s 选项时 file 命令也将去读文件列表中的块特殊文件和字符特殊文件。一般用于从原始磁盘分区中获得文件系统类型，此文件为块特殊文件。

说明

file 命令可以知道某个文件究竟是二进制（ELF 用法）的可执行文件，还是 Shell 脚本文件，或者是其他用法的文件。file 能识别的文件类型有目录、Shell 脚本、英文文本、二进制可执行文件、C 语言源文件、文本文件、DOS 的可执行文件。file 命令能探测包括图形、音频、视频等在内的多媒体文件类型。

如果我们看到一个没有扩展名的文件 TRACK11，可以使用下面的命令：

```
$ file TRACK11
track11: MP2, 112 kBits, 44.1 kHz, Stereo
```

此时系统显示这是一个立体声文件。

2.15 echo：显示文本

作用：显示文本内容。

用法：echo [-neE] [字符串]

主要选项如下。

– -n：不输出行尾的换行符。
– -e：允许对下面列出的加反斜线转义的字符进行解释。
  Ø \a：发出警告声。
  Ø \b：删除前一个字符。
  Ø \c：最后不加上换行符号。
  Ø \f：换行但光标仍旧停留在原来的位置。
  Ø \n：换行且光标移至行首。
  Ø \r：光标移至行首，但不换行。
  Ø \t：插入 tab 字符。
  Ø \v：与\f 相同。
  Ø \\：插入“\“”字符。
  Ø \nnn：插入 nnn（八进制）所代表的 ASCII 字符。
– -E：禁止对在 STRINGs 中的那些序列进行解释。
– --help：显示帮助信息。
– --version：显示版本信息。

应用实例如下。

（1）echo 显示字符串。

普通字符串可以在 echo 后直接输入字符串，但这样当要输出某些字符（如“\”）时会有问题（这种写法下，“\”是被当作继行符处理过滤掉的，要输出一个“\”必须输入“\\”，跟 C 语言中 printf 输出的要求相像），所以最好用'string'或"string"，这样即使是“\”也可以输出，方便直观。

```
#echo hello world
hello world
#echo hello\ world
hello world
#echo hello\\ world
hello\ world
#echo 'hello\\ world' 或者: echo "hello\\ world"
hello\\ world
```

（2）echo 的转义显示：加上-e 参数输出多行。

```
#echo -e 'hello\nworld'
hello
world
```

（3）输出 ASCII 字符：echo -e \NNN（NNN 为 ASCII 字符的八进制码号，不符合八进制的将会按照字面意义进行打印）。

```
#echo -e '\61 \62 \101 \141'
```

2.16 date：显示日期和时间

作用：显示或设置系统时间与日期。

用法：date [选项]显示时间用法（以“+”开头，后面接用法）

主要选项如下。

- -d,--date=STRING：显示由 STRING 指定的时间，而不是当前时间。
- -f,--file=DATEFILE：显示 DATEFILE 中每一行指定的时间，如同将 DATEFILE 中的每行作为--date 的参数。
- -I,--iso-8601[=TIMESPEC]：按照 ISO-8601 的日期/时间格式输出时间。
- TIMESPEC=`date'：（或者不指定时）仅输出日期，等于`hours'、`minutes'或`seconds'时按照指定精度输出日期及时间。
- -r,--reference=FILE：显示 FILE 的最后修改时间。
- -R,--rfc-822：根据 RFC-822 指定格式输出日期。

– -s,--set=STRING：根据 STRING 设置时间。
– -u,--utc,--universal：显示或设置全球时间（格林尼治时间）。
– --help：显示帮助文件并退出。
– --version：显示版本信息并退出。

格式 FORMAT：控制着输出格式。仅当选项指定为全球时间时本格式才有效。分别解释如下。

Ø %H：小时（以 00～23 来表示）。
Ø %I：小时（以 01～12 来表示）。
Ø %K：小时（以 0～23 来表示）。
Ø %l：小时（以 0～12 来表示）。
Ø %M：分钟（以 00～59 来表示）。
Ø %P：AM 或 PM。
Ø %r：时间（含时、分、秒，小时以 12 小时 AM/PM 来表示）。
Ø %s：总秒数。起算时间为 1970-01-01 00:00:00 UTC。
Ø %S：秒（以本地的惯用法来表示）。
Ø %T：时间（含时、分、秒，小时以 24 小时制来表示）。
Ø %X：时间（以本地的惯用法来表示）。
Ø %Z：市区。
Ø %a：星期的缩写。
Ø %A：星期的完整名称。
Ø %b：月份英文名的缩写。
Ø %B：月份的完整英文名称。
Ø %c：日期与时间。只输入 date 命令也会显示同样的结果。
Ø %d：日期（以 01～31 来表示）。
Ø %D：日期（含年、月、日）。
Ø %j：该年中的第几天。
Ø %m：月份（以 01～12 来表示）。
Ø %U：该年中的周数。
Ø %w：该周的天数，0 代表周日，1 代表周一，依此类推。
Ø %x：日期（以本地的惯用法来表示）。
Ø %y：年份（以 00～99 来表示）。
Ø %Y：年份（以 4 位数来表示）。
Ø %n：在显示时，插入新的一行。
Ø %t：在显示时，插入 tab 字符。
Ø MM：月份（必要）。
Ø DD：日期（必要）。
Ø hh：小时（必要）。

- mm：分钟（必要）。
- CC：年份的前两位数（选择性）。
- YY：年份的后两位数（选择性）。
- ss：秒（选择性）。

> **说明**
> 只有超级用户才有权限使用 date 命令设置时间，一般用户只能使用 date 命令显示时间。

应用实例如下。

（1）显示当前时间。

```
#date
2009年08月02日星期六17:19:27 CST
```

（2）显示时间后跳行，再显示当前日期。

```
# date +%T%n%D
10:19:57
03/24/10
```

（3）显示月份。

```
 # date +%B
三月
```

（4）显示日期与设定时间（12:34:56）。

```
date --date 12:34:56
2010年 03月 24日 星期三 12:34:56 CST
```

（5）设置系统当前时间（12:34:56）。

```
# date --s 12:34:56
2010年 03月 24日 星期三 12:34:56 CST
```

2.17 script：记录 Linux 会话信息

作用：用来记录执行过的所有命令及命令的输出。

用法：script [选项] 文件

主要选项如下。

- -a,--append：追加输出。

- -c,--command<命令>：运行命令而不是交互式 Shell。
- -e,--return：返回子进程的退出代码。
- -f,--flush：每次 write（写）后运行 flush（冲刷）。
- --force：即使输出文件是链接也依然使用。
- -q,--quiet：安静模式。
- -t,--timing[=<文件>]：将时间数据输出到标准错误（或文件）。
- -V,--version：输出版本信息并退出。
- -h,--help：显示此帮助并退出。

说明

许多系统管理员都知道保留一个包含各种任务、配置改变等活动日志的重要性。对一些组织而言，保留“我做了这件事”或“约翰做了那件事”的简单日志就已足够；但另一些组织则需要记录所有改变。对终端输出进行复制粘贴可能非常乏味，我们使用一个叫作 script 的鲜为人知的程序来解决这个问题，它是大多数 Linux 产品 util-linux 软件包的一部分。script 记录会话的一切内容，包括输入的内容和看到的内容，它甚至记录颜色。因此，如果命令提示符或程序输出中包含颜色，script 将记录它。

应用实例如下。

（1）使用 script。

简单执行以下命令：

```
$ script
```

在默认情况下，它向当前目录的 typescript 文件中写入内容，然后输入的一切内容都被记录到那个文件中。要向另一个文件中记录日志，只需使用 script/path/to/file 命令。完成记录后，输入 exit 退出。这个命令将关闭 script 会话并保存文件。可以使用 cat 或其他任何程序来检查日志文件。使用 script 的缺点在于，它记录所有特殊的字符，因此输入的文件中将充满控制字符和 ANSI 转义序列。可以在 script 中使用一个非常简单的 Shell 来解决这个问题：

```
$ SHELL=/bin/sh PS1="$ " script
```

使用 script 时，不要使用交互式程序或处理窗口的程序，如 vior top，它们会破坏会话的输出结果。另外，日志文件会记录使用的任何命令行程序和完成一项任务所采取的步骤。如果需要在脚本中编辑一个文件，可以考虑退出 script 会话，然后用 script -a（它在旧会话后添加新会话）对文件进行编辑后再重新启动会话。

（2）script -a filename1 -t 2>filename2。

退出命令为 exit。

若要查看记录结果，使用 cat filename 即可，filename1 和 filename2 可以是同一个文件。

2.18 apropos：搜索关键字

作用：在 whatis 资料库中搜索特定字串。

用法：apropos[选项]关键字

主要选项如下。

- -d,--debug：输出调试信息。
- -v,--verbose：输出详细的警告信息。
- -r,--regex：将每个 keyword 作为正则表达式解释。这是默认行为。每个 keyword 将匹配手册页和描述。
- -w,--wildcard：将每个 keyword 作为 Shell 样式的通配符解释。
- -e,--exact：每个 keyword 将精确匹配手册页名字和描述。
- -a,--and：只显示匹配所有 keyword 的手册页和描述。默认显示匹配任何 keyword 的项。
- -l,--long：不根据终端宽度缩减输出。
- -s section,--section section：只查找指定的手册 section。
- -m system[,...],--systems=system[,...]：用于查找其他操作系统的手册页。
- -M path,--manpath=path：指定从其他以冒号分隔的手册页层次查找。默认使用$MANPATH 环境变量。这个选项覆盖$MANPATH 的内容。
- -L locale,--locale=locale：apropos 调用 C 函数 setlocale 来得到当前本地化信息，包括$LC_MESSAGE 和$LANG。使用该选项提供一个 locale 字符串来临时更改本地化信息。
- -C file,--config-file=file：使用这个用户配置文件而不是默认的~/.manpath。
- -h,--help：打印帮助信息并退出。
- -V,--version：打印版本信息并退出。

应用实例如下。

要“查找”文件，又不知道用什么命令，可以输入下面的命令：

```
$apropos search
```

其他部分可以根据实际情况增减。

2.19 locate：搜索文件

作用：用于查找符合条件的文件，它会去保存文件与目录名称的数据库中查找符合范本样式条件的文件或目录。

用法：locate[选项]相关字

主要选项如下。

- -b,--basename：匹配唯一的路径名称的基本文件名。
- -c,--count：只显示找到条目的号码。

- -d,--database DBPATH：用 DBPATH 替代默认的数据库（/var/lib/mlocate/mlocate.db）。
- -e,--existing：只显示当前存在的文件条目。
- -L,--follow：当文件存在时跟随蔓延的符号链接（默认）。
- -h,--help：显示本帮助。
- -i,--ignore-case：匹配模式时忽略大小写区别。
- -l,--limit,-n LIMIT：限制为 LIMIT 项目的输出（或计数）。
- -m,--mmap：忽略向后兼容性。
- -P,--nofollow,-H：当检查文件时不跟随蔓延的符号链接。
- -0,--null：输出时以 NUL 分隔项目。
- -S,--statistics：不搜索项目，显示有关每个已用数据库的统计信息。
- -q,--quiet：不报告关于读取数据库的错误消息。
- -r,--regexp REGEXP：搜索基本正则表达式 REGEXP 来代替模式。
- --regex：模式是扩展正则表达式。
- -s,--stdio：忽略向后兼容性。
- -V,--version：显示版本信息。
- -w,--wholename：匹配完整路径名（默认）。

说明

有时候，明明知道某一文件或目录存在，却不知该到哪里去找到它，这时可以使用 locate 命令来搜寻文件或目录。同 find 命令相比较，locate 命令从数据库中查找，而不是每次搜索文件系统。因为是从数据库中查找，locate 的速度远远快于 find 命令。但是，使用 locate 命令查找的结果仅仅是在当前数据库中。locate 数据库是通过 cron 的日任务更新的，用户也可以手工进行，使用“locate –u;updatedb”命令完成（通常需要 root 权限）。当我们建立好这个数据库后，就可以方便地搜寻所需文件了。

应用实例如下。

（1）建立数据库。

```
locate －u
```

（2）查找所有包含字符“kde”且在目录 bin 中的文件。

```
# locate kde | grep bin
/sbin/blockdev
/usr/kerberos/bin/kdestroy
/usr/sbin/lockdev
```

2.20 rmdir：删除目录

作用：删除一个或者多个目录。

用法：rmdir [选项]...目录...

主要选项如下。

- --ignore-fail-on-non-empty：忽略任何因目录仍有数据而造成的错误。
- -p, --parents：删除指定目录后，若该目录的上层目录已变成空目录，则将其一并删除。
- -v, --verbose：显示命令执行过程。
- --help：显示此帮助信息并退出。
- --version：输出版本信息并退出。

说明

如果目录没有数据则删除该目录。rmdir 所删除的目录，每级目录中都是空的，没有其他的文件。如果任何一级目录有文件，也不能删除。

应用实例如下。

使用-p 参数删除目录时是一级一级地进行的。

```
mkdir -p  /sirdoc/redhat/
```

注意

首先在当前目录下创建目录 sirdoc，sirdoc 内又有一个子目录 redhat。

```
# rmdir -p sirdoc/redhat/
```

删除 linuxsir 目录要用到参数-p。

2.21 chattr：改变文件的属性

作用：防止关键文件被修改。

用法：chattr [选项]文件或目录

主要选项如下。

- -R：递归处理所有的文件及子目录。
- -V：详细显示修改内容，并打印输出。
- –：失效属性。
- +：激活属性。

- =：指定属性。
- A：Atime，告诉系统不要修改对这个文件的最后访问时间。
- S：Sync，一旦应用程序对这个文件执行了写操作，系统立刻把修改的结果写到磁盘中。
- a：Append Only，系统只允许在这个文件之后追加数据，不允许任何进程覆盖或截断这个文件。如果目录具有这个属性，系统将只允许在这个目录下建立和修改文件，而不允许删除任何文件。
- i：Immutable，系统不允许对这个文件进行任何修改。如果目录具有这个属性，那么任何进程只能修改目录下的文件，不允许建立和删除文件。
- D：检查压缩文件中的错误。
- d：No dump，在进行文件系统备份时，dump 程序将忽略这个文件。
- C：Compress，系统以透明的方式压缩这个文件。从这个文件读取时，返回的是解压之后的数据；而向这个文件中写入数据时，数据被压缩后才写入磁盘。
- s：Secure Delete，让系统在删除这个文件时，使用 0 填充文件所在的区域。
- u：Undelete，当一个应用程序请求删除一个文件时，系统会保留其数据块，以便以后能够恢复删除的这个文件。

说明

chattr 命令的作用很大，其中一些功能是由 Linux 内核版本来支持的。如果 Linux 内核版本低于 2.2，那么许多功能将不能实现。同样，–D 检查压缩文件中的错误，需要 2.5.19 以上内核才能支持。另外，通过 chattr 命令修改属性能够提高系统的安全性，但是它并不适合所有的目录。chattr 命令不能保护/、/dev、/tmp、/var 目录。

应用实例如下。

（1）恢复/root 目录，即子目录的所有文件。

```
# chattr -R +u/root
```

（2）用 chattr 命令防止系统中某个关键文件被修改。

在 Linux 下，有些配置文件（passwd，fatab）是不允许任何人修改的。为了防止被误删除或修改，可以设定该文件的“不可修改位（immutable）”，命令如下：

```
# chattr +i /etc/fstab
```

2.22 cksum：文件的 CRC 校验

作用：打印每个文件的 CRC 校验和及字节统计。

用法：cksum [文件]...

　　　cksum [选项]

主要选项如下。

- --help：显示此帮助信息并退出。
- --version：输出版本信息并退出。

说明

CRC（Cyclic Redundancy Check，循环冗余校验码）是常用的校验码，在早期的通信中应用广泛，因为早期的通信技术不够可靠（不可靠性的来源是由通信技术决定的，比如电磁波通信时受雷电等因素的影响），不可靠的通信就会带来“确认信息”的困惑，比如红军和蓝军通信联合进攻山下敌军的例子。第一天红军发了一条信息要蓝军第二天一起进攻，蓝军收到信息之后发一条确认信息，但是蓝军担心的是“确认信息”如果也不可靠而没有成功到达红军那里，那自己岂不是很危险？于是红军再发一条“对确认的确认信息”，但同样的问题还是不能解决，红军仍然不敢贸然行动。对通信的可靠性检查就需要“校验”，校验是从数据本身进行检查，它依靠某种数学上约定的形式进行检查，校验的结果是可靠或不可靠，如果可靠就对数据进行处理，如果不可靠就丢弃重发或者进行修复。

CRC 码由两部分组成，前半部分是信息码，就是需要校验的信息，后半部分是校验码。如果 CRC 码共长 n bit，信息码长 k bit，就称为(n,k)码。它的编码规则如下。

（1）首先将原信息码（$k$bit）左移 r 位（$k+r=n$）。

（2）然后运用一个生成多项式 g(x)（也可看成二进制数）用模 2 除上面的式子，得到的余数就是校验码。

要说明的是，模 2 除就是在除的过程中用模 2 加，模 2 加实际上就是我们熟悉的异或运算，就是加法不考虑进位，公式如下。

$$0+0=1+1=0,1+0=0+1=1$$

即“异”则真，“非异”则假。

由此得到定理：

$$a+b+b=a$$

也就是“模 2 减”和“模 2 加”真值表完全相同。

有了加减法就可以用来定义模 2 除法，于是就可以用生成多项式 $g(x)$生成 CRC 校验码。

例如，$g(x)=x^4+x^3+x^2+1$,（7,3）码，信息码 110 产生的 CRC 码就是：11101 | 110,0000（设 a=11101，b=1100000）。

取 b 的前 5 位 11000 跟 a 异或得到 101，101 加上 b 没有取到的 00 得到 10100，然后跟 a 异或得到 01001，也就是余数 1001，余数是 1001，所以 CRC 码是 110,1001。

对于 $g(x)=x^4+x^3+x^2+1$ 的解释：（都是从右往左数）x^4 表示第 5 位是 1，因为没有 x^1，所以第 2 位就是 0。

标准的 CRC 码是 CRC-CCITT 和 CRC-16，它们的生成多项式如下。

$$CRC\text{-}CCITT=x^{16}+x^{12}+x^{5}+1$$

$$CRC\text{-}16=x^{16}+x^{15}+x^{2}+1$$

应用实例如下。

使用 CRC 检验文件，代码如下：

```
cksum 1.txt
2422907486 124 1.txt
```

2.23 cmp：比较文件差异

作用：比较文件差异。

用法：cmp [选项] [第 1 个文件] [第 2 个文件]

主要选项如下。

- -c--print-chars：除了标明差异处的十进制字码之外，一并显示该字码所对应的字符。
- -i<字符数目>：--ignore-initial=<字符数目>：指定一个数目。
- -l--verbose：标识出所有不一样的地方。
- -s--quiet 或--silent：不显示错误信息。
- -v--version：显示版本信息。
- --help：在线帮助。

函数的返回值如下。

- 0：文件是同样的。
- 1：文件是不同的。
- >1：发生错误。

应用实例如下。

比较两个文本文件，代码如下：

```
# cmp hello.cpp hello1.cpp
hello.cpp hello1.cpp differ: byte 9, line 1
```

2.24 split：分割文件

作用：分割文件。

用法：split [选项]... [输入[前缀]]

主要选项如下。

- -d：数字表示切割后的小文件。
- -v：显示详细的处理信息。
- -b<字节>：每个分割文件的大小。
- -c<字节>：输出每行的大小。
- -l<行数>：指定行数。
- --help：显示帮助信息。
- --version：显示版本信息。
- [PREFIX]：设置分割后文件的前置文件名，split 会自动在前置文件名后加上编号。

应用实例如下。

把一个大文件 file 分割成 1MB 大小的文件。

```
Split -b 1m file filebak_
```

2.25 dirname：显示文件除名字外的路径

作用：显示文件除名字外的路径。

用法：dirname 名称

dirname 选项

主要选项如下。

- --help：显示此帮助信息并退出。
- --version：输出版本信息并退出。

应用实例如下。

（1）查看/usr/bin/sort 路径。

```
# dirname /usr/bin/sort
/usr/bin
```

（2）查看 stdio.h 路径。

```
dirname stdio.h
```

2.26 find：查找目录或者文件

作用：查找文件或目录。

使用权限：所有用户。

用法：find [-H] [-L] [-P] [-Olevel] [-D help|tree|search|stat|rates|opt|exec] [path...] [expression]

主要选项如下。

- path...路径：告诉 find 在哪儿去找用户要的东西。
- -exec find：对匹配的文件执行该参数所给出的其他 linux 命令。相应命令的形式为' 命令 - and' {} \;，注意{ }和\;之间的空格。
- -ok：和- exec 的作用相同，只不过和会人交互而已，OK 执行前会向用户确认是不是要执行。

find 命令的主要参数如下。

- -name：按照文件名查找文件。
- -perm：按照文件权限查找文件。
- -prune：使用这一选项可以使 find 命令不在当前指定的目录中查找，如果同时使用了-depth 选项，那么-prune 选项将被 find 命令忽略。
- -user：按照文件属主查找文件。
- -group：按照文：所属的组查找文件。
- -mtime -n +n：按照文件的更改时间查找文件，-n 表示文件更改时间距现在 *n* 天以内，+n 表示文件更改时间距现在 *n* 天以前。find 命令还有-atime 和-ctime 选项，但它们都和-mtime 选项相似，所以在这里只介绍-mtime 选项。
- -nogroup：查找无有效所属组的文件，即该文件所属的组在/etc/groups 中不存在。
- -nouser：查找无有效属主的文件，即该文件的属主在/etc/passwd 中不存在。
- -newer file1 ! file2：查找更改时间比文件 file1 新但比文件 file2 旧的文件。
- -type：查找某一类型的文件。
  - b：块设备文件。
  - d：目录。
  - c：字符设备文件。
  - p：管道文件。
  - l：符号链接文件。
  - f：普通文件。
  - s：socket 文件。
- -size n[c]：查找文件长度为 *n* 块的文件，带有 c 时表示文件长度以字节计。
- -depth：在查找文件时，首先查找当前目录中的文件，然后再在其子目录中查找。
- -fstype：查找位于某一类型文件系统中的文件，这些文件系统类型通常可以在配置文件/etc/fstab 中找到，该配置文件中包含了本系统中有关文件系统的信息。
- -mount：在查找文件时不跨越文件系统 mount 点。
- -follow：如果 find 命令遇到符号链接文件，就跟踪至链接所指向的文件。
- -cpio：对匹配的文件使用 cpio 命令，将这些文件备份到磁带设备中。

应用实例如下。

find 命令查找文件的 5 种方法如下。

（1）文件名查找法。

```
find / -name named.conf
```

（2）快速查找文件法。

如果知道文件存放在某个目录中，那么只要在这个目录中往下寻找就能节省很多时间。比如 named.conf 文件，从它的文件扩展名“.conf”可以判断这是一个配置文件，那么它应该在/etc 目录内，此时可以使用下面的命令：

```
find /etc -name named.conf
```

（3）根据部分文件名查找法。

有时仅知道某个文件包含有 abdd 这 4 个字符，那么要查找系统中所有包含这 4 个字符的文件，可以输入下面的命令：

```
find / -name '*abdd*'
```

输入这个命令以后，Linux 系统将会在“/”目录中查找所有包含 abdd 这 4 个字符的文件（其中“*”是通配符），比如 abddrmyz 等符合条件的文件都能显示出来。

（4）根据文件的特征查找法。

有时知道某个文件的大小、修改日期、所属用户等特征，也可以使用“find”命令查找出文件来。例如，查找在系统中属于已经作废用户的文件，可以使用下面的命令：

```
find  / -nouser
```

（5）混合查找方式查找法。

find 命令可以使用混合查找的方法。例如，要想在“/home”目录中查找大于 500 000 字节，并且属于 cao 用户的文件，可以使用-and（与）把两个查找参数连接起来组合成一个混合的查找方式。

```
find  /home -size +500000c -and -user cao
```

2.27 findfs：通过列表或用户 ID 查找文件系统

作用：用来查找指定卷标的文件系统。

用法：findfs LABEL=<label>

findfs UUID=<uuid>

主要选项如下。

- LABEL=<label>：卷标名称。
- UUID=<uuid>：分区的 UUID 号。

说明

随着 Linux 系统中硬盘容量和数目的增加，Linux 系统中的分区数量也越来越多，使用 findfs 命令可以通过卷标名称或 UUID 号快速定位分区位置。

应用实例如下。

查找卷标名称是“/var/ftp”的分区位置，使用如下命令：

```
#findfs LABEL=/var/ftp
/dev/hda9
```

2.28 ln：链接文件或目录

作用：链接文件或目录。

用法：ln [选项] [源文件或目录] [目标文件或目录]

　　　ln [选项] [源文件或目录...] [目的目录]

主要选项如下。

- -b--backup：删除、覆盖目标文件之前的备份。
- -d-F--directory：建立目录的硬链接。
- -f--force：强行建立文件目录的链接，不论文件目录是否存在。
- -i--interactive：覆盖既有文件之前先询问用户。
- -n--no-dereference：把符号链接的目的目录视为一般文件。
- -s--symbolic：对源文件建立符号链接，而非硬链接。
- -S<字尾备份字符串>--suffix=<字尾备份字符串>：用“-b”参数备份目标文件后，备份文件的字尾会被加上一个备份字符串，预设的字尾备份字符串是符号“～”，可通过“-S”参数来改变它。
- -v--verbose：显示命令执行过程。
- -V<备份方式>--version-control=<备份方式>：用“-b”参数备份目标文件后，备份文件的字尾会被加上一个备份字符串，这个字符串不仅可用“-S”参数变更，当使用“-V”参数<备份方式>指定不同的备份方式时，也会产生不同字尾的备份字符串。
- --help：在线帮助。
- --version：显示版本信息。

说明

ln 命令用于链接文件或目录。如果同时指定两个以上的文件或目录，且最后的目的地是一个已经存在的目录，则会把前面指定的所有文件或目录复制到该目录中。若同时指定多个文件或目录，且最后的目的地并非一个已存在的目录，则会出现错误信息。

链接有两种，一种被称为硬链接（Hard Link），另一种被称为符号链接（Symbolic Link）。建立硬链接时，链接文件和被链接文件必须位于同一个文件系统中，并且不能建立指向目录的硬链接。而对于符号链接，则不存在这个问题。默认情况下，ln 产生硬链接。

在硬链接的情况下，参数中的“目标”被链接至[链接名]。如果[链接名]是一个目录名，系统将在该目录之下建立一个或多个与“目标”同名的链接文件，链接文件和被链接文件的内容完全相同。如果[链接名]是一个文件，用户将被告知该文件已存在且不进行链接。如果指定了多个“目标”参数，那么最后一个参数必须为目录。

应用实例如下。

（1）为当前目录下的文件 lunch 创建一个符号链接/home/xu。

```
$ ln - s lunch /home/xu
```

说明

给 ln 命令加上-s 选项，则建立符号链接。如果[链接名]已经存在，但不是目录，将不进行链接。[链接名]可以是任何一个文件名（可包含路径），也可以是一个目录，并且允许它与“目标”不在同一个文件系统中。如果[链接名]是一个已经存在的目录，系统将在该目录下建立一个或多个与“目标”同名的文件，此新建的文件实际上是指向原“目标”的符号链接文件。

（2）创建到一个文件的另一个链接。

```
#ln    -f chap1    intro
```

这会将 chap1 链接到新的名称 intro。如果 intro 不存在，则会创建该文件名。如果 intro 已经存在，那么这个文件会被替换为指向 chap1 的一个链接，然后 chap1 和 intro 文件名会指向同一个文件。对这里面任何一个文件的更改都会出现在另一个文件中。如果一个文件名被 rm 命令删除，那么该文件并没有完全被删除，因为它依旧以其他名字存在。

（3）将文件链接为另一个目录中的相同名字。

```
#ln    index    manual
```

这会将 index 链接到新名称 manual/index。

（4）将几个文件链接为另一个目录中的名称。

```
ln    chap2    jim/chap3    /home/manual
```

这会将 chap2 链接到新名称/home/manual/chap2，将 jim/chap3 链接到新名称/home/manual/chap3。

（5）在不指明目标文件参数的情况下得到相同的结果。

```
ln -s /tmp/toc
```

2.29 lndir：链接目录内容

作用：链接目录内容。

用法：lndir [-ignorelinks] [-silent] [源目录] [目的目录]

主要选项如下。

- -ignorelinks：直接建立符号链接的符号链接。
- -silent：不显示命令执行过程。

> **说明**
>
> 执行 lndir 命令，可一口气把源目录下的文件和子目录统统建立起相互对应的符号链接。

2.30 lsattr：显示文件属性

作用：查看由 chattr 设置的文件属性。

用法：lsattr [选项] [文件或者目录]

主要选项如下。

- -R：递归地列出目录及其下内容的属性。
- -V：显示程序版本。
- -a：列出目录中的所有文件，包括以“.”开头的文件的属性。
- -d：以列出其他文件的方式列出目录的属性，而不列出其下的内容。
- -v：显示文件版本。

应用实例如下。

（1）查看一个 sbin 目录的属性而不是内容。

```
# lsattr -d /sbin
----------I-- /sbin
```

（2）递归地列出/home/cjh 目录及其下内容的属性。

```
# lsattr -R /home/cjh
------------- /home/cjh/1
------------- /home/cjh/ch
/home/cjh/ch:
```

2.31 od：输出文件内容

作用：od 命令会读取所给予的文件的内容，并将其内容以八进制字码呈现出来。

用法：od [选项]...[文件]...

主要选项如下。

将指定文件以八进制形式（默认）转储到标准输出。如果指定了多于一个的文件参数，程序会自动将输入的内容整合为列表并以同样的形式输出。

如果没有指定文件，或指定文件为“-”，程序将从标准输入读取数据。

长选项必须用的参数在使用短选项时也是必需的。

- -A,--address-radix=初始偏移：设置要显示的文件的初始偏移量。
- -j,--skip-bytes=字节数：略过指定字节数的输入内容。
- -N,--read-bytes=字节数：限制要转储的输入字节为指定的字节数。
- -S,--strings[=字节数]：输出至少包含指定字节数的字符相应的字串。
- -t,--format=类型：选取输出格式类型。
- -v,--output-duplicates：不使用“*”字符标记屏蔽的行。
- -w,--width[=字节数]：每一行只输出指定的字节数。
- --traditional：接受传统形式提交的参数。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

传统格式命令可混合使用，不同的格式可叠加。

- -a：即-ta，使用命名的字符，忽略高阶位。
- -b：即-to1，使用八进制单字节字符。
- -c：即-tc，使用 ASCII 字符，或者反斜杠文件名特殊字符。
- -d：即-tu2，使用无符号十进制双字节字符。
- -f：即-tfF，指定浮点数的对照输出格式。
- -i：即-tdl，指定十进制整数的对照输出格式。
- -l：即-tdL，指定十进制长整数的对照输出格式。
- -o：即-to2，指定双字节单位八进制数的对照输出格式。
- -s：即-td2，指定双字节单位十进制数的对照输出格式。
- -x：即-tx2，指定双字节单位十六进制数的对照输出格式。

如果两个调用格式都适用，系统会在最后一个运算量以“+”或数字开始（如果有两个运算量）的情况下采用第二种格式。

偏移量是指 -j 偏移量。标签是第一个字节的伪地址，随着累积过程递增。偏移量和标签如果冠以“0x”或“0X”前缀则表示十六进制数；后缀“.”代表八进制数，后缀“b”表示乘以 512。

TYPE 是由下面一种或几种参数组成的：

- a：重定义字符，忽略高位字节。

– c：ASCII 码字符或者反斜线字串（如\n）。
– d[尺寸]：有符号十进制数，每个整型数占指定尺寸的字节。
– f[尺寸]：浮点数，每个整型数占指定尺寸的字节。
– o[尺寸]：八进制数，每个整型数占指定尺寸的字节。
– u[尺寸]：无符号十进制数，每个整型数占指定尺寸的字节。
– x[尺寸]：十六进制数，每个整型数占指定尺寸的字节。

尺寸应该是一个数字。对 doux 等类型来说，适用的尺寸有：C 代表 sizeof(char)，S 代表 sizeof(short)，I 代表 sizeof(int)，或 L 代表 sizeof(long)；如果类型是 f，可适用的尺寸有：F 代表 sizeof(float)，D 代表 sizeof(double)，或 L 代表 sizeof(long double)。

当 RADIX 为“d”时代表十进制，为“o”时代表八进制，为“x”时代表十六进制，“n”则代表无。

BYTES 是以 0x 或 0X 为前缀的十六进制数，可能有以下几种后缀：

b 512，kB 1000，K 1024，MB 1000×1000，M 1024×1024，GB 1000×1000×1000，G 1024×1024×1024，对于 T、P、E、Z、Y 同样适用。

为任何类型添加“z”后缀都将在每个输出行的行尾显示可显示字符。

不带数字的--string 选项代表 3；不带数字的--width 选项代表 32。

默认情况下 od 使用-A o -t oS -w16 参数。

应用实例如下。

显示一个.txt 文件的八进制码。

```
# od 1.txt
0000000 072560 066142 061551 061440 060554 071563 044040 066145
0000020 067554 075440 005012 072560 066142 061551 044040 066145
0000040 067554 024450 075440 076412 005012 072560 066142 061551
0000060 071440 060564 064564 020143 067566 062151 066440 064541
0000160 062154 021041 035451 076412 005012 005175
0000174
```

2.32 paste：合并文件的列

作用：paste 命令会把每个文件以列对列的方式一列列地加以合并。

用法：paste [选项]...[文件]...

主要选项如下。

– -d,--delimiters=列表：改用指定列表里的字符替代制表分隔符。
– -s,--serial：不使用平行的行目输出模式，而是每个文件占用一行。
– --help：显示此帮助信息并退出。
– --version：显示版本信息并退出。

应用实例如下。

显示合并一个.txt 文件的列。

```
# paste 1.txt
public class Hello {
public Hello() {
}
public static void main(String[] args) {
System.out.println("Hello World! ");
}
}
```

2.33 stat：显示 inode 内容

作用：stat 以文字的用法来显示 inode 的内容。

用法：stat [文件或目录]

　　stat [选项]...文件...

显示文件或文件系统的状态。

- -L,--dereference：跟随链接。
- -f,--file-system：显示文件系统状态而非文件状态。
- -c--format=格式：使用指定输出格式代替默认值，每用一次指定格式换一新行。
- --printf=格式：类似--format，但是会解释反斜杠转义符，不使用换行作输出结尾。如果仍希望使用换行，可以在格式中加入“\n”。
- -t,--terse：使用简洁格式输出。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

有效的文件格式序列（不使用--file-system）如下。

- %a：八进制权限。
- %A：用可读性较好的方式输出权限。
- %b：计算已分配块数（参见%B）。
- %B：以字节为单位输出%b 所报告的每个块的大小。
- %C：SELinux 安全环境字符串。
- %d：十进制设备编号。
- %D：十六进制设备编号。
- %f：十六进制原始模式。
- %F：文件类型。
- %g：文件的属组 ID。

- %G：文件的属组组名。
- %h：硬链接数量。
- %i：Inode 编号。
- %m：挂载点。
- %n 文件名。
- %N：如果对象是一个符号链接，则显示引用到的其他文件名。
- %o：I/O 块大小。
- %s：总计大小，以字节为单位。
- %t：十六进制主设备类型。
- %T：十六进制子设备类型。
- %u：文件的属主 ID。
- %U：文件的属主用户名。
- %w：文件创建时间，若未知则显示“-”。
- %W：从 UNIX 元年起以秒计的文件创建时间，若未知则显示“-”。
- %x：上次访问时间。
- %X：从 UNIX 元年起以秒计的上次访问时间。
- %y：上次修改时间。
- %Y：从 UNIX 元年起以秒计的上次修改时间。
- %z：上次更改时间。
- %Z：从 UNIX 元年起以秒计的上次更改时间。

有效的文件系统格式序列如下。

- %a：非超级用户可用的剩余块数。
- %b：文件系统的总数据块数。
- %c：文件系统中文件节点总数。
- %d：文件系统中空闲文件节点数。
- %f：文件系统中空闲块数。
- %i：十六进制文件系统 ID。
- %l：允许的文件名最大长度。
- %n：文件名。
- %s：块大小（用于快速传输）。
- %S：基本块大小（用于块计数）。
- %t：十六进制类型描述。
- %T：可读性较好的类型描述。

应用实例如下。

查看.txt 文件的 inode 内容。

```
# stat 1.txt
  File: “1.txt”
  Size: 0              Blocks: 0            IO Block: 4096   一般空文件
```

```
Device: fd00h/64768d    Inode: 1182306     Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)  Gid: (    0/    root)
Access: 2008-08-02 20:11:28.000000000 +0800
Modify: 2008-08-02 20:11:27.000000000 +0800
Change: 2008-08-02 20:11:27.000000000 +0800
```

说明

inode 译成中文就是索引节点。每个存储设备或存储设备的分区(存储设备是硬盘、软盘、U 盘……)被用法化为文件系统后，应该有两部分，一部分是 inode，另一部分是 Block。Block 是用来存储数据的。而 inode 用来存储这些数据的信息，包括文件大小、属主、归属的用户组、读/写权限等。inode 为每个文件进行信息索引，所以就有了 inode 的数值。操作系统根据命令，能通过 inode 值最快地找到相对应的文件。

2.34 tee：读取标准输入到标准输出并可保存为文件

作用：读取标准输入的数据，并将其内容输出成文件。

用法：tee [选项] [文件...]

主要选项如下。

- -a,--append：内容追加到给定的文件而非覆盖。
- -i,--ignore-interrupts：忽略中断信号。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

如果文件指定为“-”，则将输入内容复制到标准输出。

说明

tee 命令会从标准输入设备读取数据，将其内容输出到标准输出设备，同时保存成文件。

应用实例如下。

（1）要同时查看和保存一个命令的输出。

```
#sort program.c | tee program.lint
```

它在工作站上显示命令 sort program.c 的标准输出，同时在文件 program.lint 中保存输出的一个副本。如果 program.lint 文件早已存在，它将被删除并替换。

（2）要同时查看一个命令的输出并保存到一个现有文件。

```
# sort program.c | tee  -a program.lint
```

它将在工作站上显示 sort program.c 命令的标准输出，同时在 program.lint 文件尾部添加输出的一个副本。如果 program.lint 文件不存在，它将被创建。

2.35 tmpwatch：删除临时文件

作用：执行 tmpwatch 命令可删除不必要的暂存文件，可以设置文件超期时间，单位以小时计算。

用法：tmpwatch [选项] [超期时间] [目录...]

主要选项如下。

- -a--all：删除任何类型的文件。
- -f--force：强制删除文件目录，其效果类似于 rm 命令的“-f”参数。
- -q--quiet：不显示命令执行过程。
- -v--verbose：详细显示命令执行过程。
- -test：仅做测试，并不真正删除文件目录。

应用实例如下。

（1）删除/tmp 目录下 7 天内没有访问的文件。

```
tmpwatch 7*24 /tmp
```

（2）删除/tmp 目录下 100 小时内没有使用的文件。

```
#tmpwatch 100 /tmp
```

2.36 touch：更新文件目录时间

作用：修改文件时间信息。

用法：touch [选项]...[目录]...

主要选项如下。

- -a：改变文件的读取时间记录。
- -c, --no-create：不建立任何文件。
- -d, --date=STRING：设定时间与日期，可以使用各种不同的用法。
- -F SEC, --forward=SEC：使用参考文件的时间记录。
- -m：改变文件的修改时间记录。

– -r, --reference=文件：使用指定文件的时间属性而非目前的时间。
– -t STAMP：使用[[CC]YY]MMDDhhmm[.ss]用法的时间而非目前的时间。

应用实例如下。

（1）将文件的时间记录改为现在的时间。

```
touch myfile
```

若文件不存在，系统会建立一个新的文件。

（2）将文件 myfile 的时间记录改为 1 月 6 日 18∶03。

```
touch -c -t 01061803 myfile
```

时间的用法可以参考 date 命令，至少需要输入 MMDDhhmm，也就是月、日、时与分。

（3）将 myfile 的时间记录改成公元 2005 年 1 月 6 日 18∶03。

```
touch -d "6:03pm 01/06/2005" file
```

时间可以使用 am、pm，是 24 小时的用法。日期可以使用其他用法，如 6 JAN 2005。

（4）应用进阶。

有时 Linux 文件系统空间被占满了，要查询滥用磁盘的用户。首先要查出哪些文件是新创建的、哪些是更新的，以便找出不正常的大文件。使用命令 touch 生成一个文件，时间戳会根据情况设定为比较近的日期，再使用 find 命令找出更新的文件。

```
touch -d "6:03pm 02/06/2005" test
find /home -newer test –print
```

系统会在屏幕上显示那些更新的文件的位置，也就知道了它的用户。

2.37 tree：以树状图显示目录内容

作用：有时候我们想了解一个文件夹或者驱动器根目录下的所有文件，并希望它以资源管理器的树形视图方式显示文件结构。

用法：tree [选项] [目录...]

主要选项如下。

– -a：显示所有文件和目录。
– -A：使用 ASNI 绘图字符显示树状图而非以 ASCII 字符组合。
– -C：给文件和目录清单加上色彩，便于区分各种类型。
– -d：显示目录名称而非内容。
– -D：列出文件目录的更改时间。
– -f：在每个文件目录之前，显示完整的相对路径名称。
– -F：在执行文件、目录、Socket、符号链接、管道名称前各自加上“*”、“/”、“=”、“@”、“|”号。

- -g：列出文件目录的所属群组名称，没有对应的名称时，则显示群组识别码。
- -i：不以阶梯状列出文件目录名称。
- -I<范本样式>：不显示符合范本样式的文件目录名称。
- -l：如遇到性质为符号链接的目录，直接列出该链接所指向的原始目录。
- -n：不给文件和目录清单加上色彩。
- -N：直接列出文件和目录名称，包括控制字符。
- -p：列出权限标识。
- -P<范本样式>：只显示符合范本样式的文件目录名称。
- -q：用“?”取代控制字符，列出文件和目录名称。
- -s：列出文件目录大小。
- -t：按文件和目录的更改时间排序。
- -u：列出文件目录的拥有者名称，没有对应的名称时，则显示用户识别码。
- -x：将范围局限在现行的文件系统中。若指定目录下的某些子目录存放于另一个文件系统上，则将该子目录排除在寻找范围外。

应用实例如下。

查看/root/ossec-hids-1.5 目录树结构，分页显示。

```
# tree /root/ossec-hids-1.5 |more
/root/ossec-hids-1.5
|-- BUGS
|-- CONFIG
|-- CONTRIB
|-- INSTALL
|-- LICENSE
|-- README
|-- active-response
|   |-- disable-account.sh
|   |-- firewall-drop.sh
|   |-- firewalls
|   |   |-- ipfw.sh
|   |   |-- ipfw_mac.sh
|   |   `-- pf.sh
|   |-- host-deny.sh
|   `-- route-null.sh
```

2.38 umask：指定在建立文件时预设的权限掩码

作用：指定在建立文件时预设的权限掩码。

用法：umask [-p] [-S] [mode]

主要选项如下。

- -S：以文字的方式表示权限掩码。
- -p：以数字的方式表示权限掩码。
- mode：权限掩码。

说明

当最初登录到系统中时，umask 命令就确定了创建文件的默认模式。这一命令实际上和 chmod 命令正好相反。系统管理员必须要为用户设置一个合理的 umask 值，以确保用户创建的文件具有所希望的默认权限，防止其他非同组用户对用户的文件具有写权限。在已经登录之后，可以按照个人的偏好使用 umask 命令来改变文件创建的默认权限。相应的改变直到退出该 Shell 使用另外的 umask 命令之前一直有效。一般来说，umask 命令是在/etc/profile 文件中设置的，每个用户在登录时都会引用这个文件，所以如果希望改变所有用户的 umask，可以在该文件中加入相应的条目。如果希望永久性地设置自己的 umask 值，那么就把它放在自己的$HOME 目录下的.profile.bash_profile 文件中。

umask 命令允许设定文件创建时的默认模式，对应每一类用户（文件属主、同组用户、其他用户）存在一个相应的 umask 值中的数字。对于文件来说，这一数字的最大值是 6。系统不允许在创建一个文本文件时就赋予它执行权限，必须在创建后用 chmod 命令增加这一权限。目录则允许设置执行权限，这样针对目录来说，umask 中各个数字最大可以是 7。

表 2-1 列出了常用的 umask 值及对应的目录和文件权限。

表 2-1　常用的 umask 值及对应的目录和文件权限

umask 值	目录权限	文件权限
022	775	644
027	750	640
002	755	664
006	771	660
007	770	660

应用实例如下。

（1）如果想知道当前的 umask 值，可以使用没有任何参数的 umask 命令。

```
#umask
0022
```

（2）如果想要改变 umask 值，只要使用 umask 命令设置一个新值即可。

```
#umask 002
```

（3）确认系统是否已经接受了新的 umask 值。

```
# umask 002
# umask
0002
```

在使用 umask 命令之前，一定要弄清楚到底希望文件/目录具有什么样的默认权限，否则可能会得到一些非常奇怪的结果。例如，如果将 umask 值设置为 600，那么所创建的文件/目录的默认权限就是 066。

（4）使用 umask 命令设置安全权限。

```
#  umask 117
#  umask -S
u=rw,g=rw,o=
```

上述命令把 umask 值改为 117，结果只有文件所有者具有读/写文件的权限，其他用户不能访问该文件。这显然是一种非常安全的状态。

2.39 chmod：设置文件或目录的访问权限

作用：chmod 是 change file or directory’s mode 的缩写，其作用是修改文件或目录的访问权限，用户可以用它控制文件或目录的访问权限。

用法：chmod 命令有两种用法，一种是包含字母和操作符表达式的字符设定法（相对权限设定），另一种是包含数字的数字设定法（绝对权限设定）。

1. 字符设定法

字符设定法的一般形式如下：

```
chmod [ugoa] [+ | — | =] [rwxXstugo] 文件名
```

第一部分决定权限的授予者，第二部分决定对权限进行何种操作（添加、删除、设定），第三部分决定具体要授予的权限，3 部分组成一个字符串，也就是 mode 参数值。详细描述如表 2-2 所示。

表 2-2　chmod 字符设定模式的用法

值	描　述
授予者	
u	用户，即文件或目录的所有者
g	同组用户，与文件属主有相同组 ID 的所有用户

续表

值	描　述
o	其他用户
a	所有用户，它是系统默认值
操作符号	
＋	在原有权限中添加某个权限
－	在原有权限中删除某个权限
＝	赋予新给定的权限，并取消其他所有权限
权限位	
r	设置读权限
w	设置写权限
x	设置执行权限
X	只有目标文件对某些用户是可执行的，该目标文件是目录时才追加 X 属性
s	设置 SUID 和 SGID，g+s 设置 SGID，u+s 设置 SUID
t	保存程序的文本到交换设备上
u	与文件属主拥有一样的权限
g	与和文件属主同组的用户拥有一样的权限
o	与其他用户拥有一样的权限

应用实例如下。

如果一个系统管理员写了一则通知（news）让所有用户阅读，那么必须授权用户对这个文件有读权限，可以使用如下命令：

```
#chmod a＝r news
```

2．数字设定法

数字设定法的一般形式如下：

```
chmod[mode]文件名
```

数字属性的用法应为 3 个 0～7 之间的八进制数，其顺序是（u）（g）（o）文件名，以空格分开要改变权限的文件列表，支持通配符。

数字表示的权限的含义如下。

- 0001：所有者的执行权限。
- 0002：所有者的写权限。
- 0004：所有者的读权限。
- 0010：组的执行权限。
- 0020：组的写权限。
- 0040：组的读权限。
- 0100：其他人的执行权限。
- 0200：其他人的写权限。

– 0400：其他人的读权限。
– 1000：粘贴位置位。
– 2000：假如这个文件是可执行文件，则为组 ID 位置位。
– 4000：假如这个文件是可执行文件，则为用户 ID 位置位。

系统管理员写了一则通知（news2）让所有用户阅读，那么必须授权用户对这个文件有读权限，可以使用以下命令：

```
#chmod 444 news2
```

上述命令中，数字 444 是如何计算出来的呢？0004 为所有者的读权限，0040 为组的读权限，0400 为其他人的读权限，这 3 个数字相加就是 0444（以上数字都是八进制数），如图 2-6 所示。

```
[root@localhost tmp]# chmod a=r news
[root@localhost tmp]# chmod 444 news1
[root@localhost tmp]# ls -l
total 0
-r--r--r-- 1 root root 0 Mar 24 13:03 news
-r--r--r-- 1 root root 0 Mar 24 13:03 news1
[root@localhost tmp]# _
```

图 2-6　分别用字符设定法和数字设定法修改文件权限

从图 2-6 中可以看到，“chmod 444 news1”和“chmod a=r news”命令是等价的。

说明

因为 Linux 系统有能力支持多用户，在每一方面系统都会作出谁能读、写和执行的资源权力限制。这个权限以 3 个 8 位元的方式存储，一个表示文件所属者，一个表示文件所属群组，另一个表示其他人。这些数字的表示形式如表 2-3 所示。

表 2-3　权限的表示形式

数　字	二进制数的表示	对应的权限表示形式	权限含义
0	000	---	没有任何权限
1	001	--x	可执行
2	010	-w-	可写
3	011	-wx	可写和可执行
4	100	r--	可读
5	101	r-x	可读和可执行
6	110	rw-	可读和可写
7	111	rwx	可读、可写和可执行

2.40 chgrp：改变文件或目录所属的群组

作用：chgrp 是 change group permissions 的缩写，chgrp 是一条在 Linux 和 UNIX 中用于设置文件所归属的组的命令。与 chown 命令不同，chgrp 允许普通用户改变文件所属的组，只要该用户是该组的一员。

用法：chgrp [选项]…组文件…

　　chgrp [选项]…--reference=参考文件…

主要选项如下。

- -c, --changes：类似--verbose 选项，只在有更改时才显示结果。
- --dereference：影响符号链接所指示的对象，而非符号链接本身。
- -h, --no-dereference：影响符号链接本身，而非符号链接所指示的目的地（当系统支持更改符号链接的所有者时，此选项才有效）。
- -f, --silent：不显示错误信息。
- --reference=参考文件：使用<参考文件>的所属组，而非指定的<组>。
- -R, --recursive：递归处理所有的文件及子目录。
- -v, --verbose：显示命令执行过程。

> **说明**
>
> chgrp 命令改变指定文件所属的用户组。其中 group 可以是用户组 ID，也可以是/etc/group 文件中用户组的组名。文件名是以空格分开的要改变所属组的文件列表，支持通配符。如果用户不是该文件的属主、超级用户，则不能改变该文件的组。

应用实例如下。

将目录 dir 中的所有文件属组改为 sam。

```
# chgrp -R sam dir/
```

2.41 chown：改变文件的拥有者或群组

作用：chown 是 change ownership 的缩写，用于更改与文件关联的所有者组。

用法：chown [选项]…所有者[:[组]]文件…

　　chown [选项]…：组文件…

　　chown [选项]…--reference=参考文件…

主要选项如下。

当使用--referebce 参数时，将文件的所有者和所属组更改为与指定参考文件相同。

- -c,--changes：类似 verbose，但只在有更改时才显示结果。

- --dereference：受影响的是符号链接所指示的对象，而非符号链接本身。
- -h,--no-dereference：会影响符号链接本身，而非符号链接所指示的目的地（当系统支持更改符号链接的所有者时，此选项才有效）。
- --from=当前所有者:当前所属组：只当每个文件的所有者和组符合选项所指定时才更改所有者和组。其中一个可以省略，这时已省略的属性就不需要符合原有的属性。
- --no-preserve-root：不特殊对待“/”（默认值）。
- --preserve-root：不允许在“/”上递归操作。
- -f,--silent,--quiet：去除大部分的错误信息。
- --reference=参考文件：使用参考文件的所属组，而非指定值。
- -R,--recursive：递归处理所有的文件及子目录。
- -v,--verbose：为处理的所有文件显示诊断信息。

以下选项是在指定了-R 选项时被用于设置如何穿越目录结构体系。如果用户指定了多于一个选项，那么只有最后一个选项会生效。

- -H：如果命令行参数是一个通到目录的符号链接，则遍历符号链接。
- -L：遍历每一个遇到的通到目录的符号链接。
- -P：不遍历任何符号链接（默认）。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

如果没有指定所有者，则不会更改。所属组若没有指定也不会更改，但当加上“:”时 GROUP 会更改为指定所有者的主要组。所有者和所属组可以是数字或名称。

说明

chown 将指定文件的拥有者改为指定的用户组，用户可以是用户名用户 ID；组可以是组名组 ID；文件是以空格分开的要改变权限的文件列表，支持通配符。系统管理员经常使用 chown 命令，在将文件复制到另一个用户的目录下以后，让用户拥有使用该文件的权限。

应用实例如下。

将目录/mywork 及其下的所有文件和子目录的属主改为 lwan，属组改为 users。

```
# chown - R lwan.users /mywork
```

2.42 more：查看文件的内容

作用：和 cat 命令相比，more 命令适合查看大文件。more 命令一般用于显示内容超过一屏的文件。其他命令经常和 more 匹配使用，但 more 命令也可单独使用。

用法：more [选项]文件

主要选项如下。

- -d：显示帮助，而不是响铃。
- -f：统计逻辑行数而不是屏幕行数。
- -l：抑制换页（form feed）后的暂停。
- -p：不滚屏，清屏并显示文本。
- -c：不滚屏，显示文本并清理行尾。
- -u：抑制下画线。
- -s：将多个空行压缩为一行。
- -NUM：指定每屏显示的行数为 NUM。
- +NUM：从文件第 NUM 行开始显示。
- +/STRING：从匹配搜索字符串 STRING 的文件位置开始显示。
- -V：输出版本信息并退出。

more 命令在命令暂停时接受子命令并且作为-p 标志的参数。许多子命令接受可选的整数，这里由 K 代表，它必须在子命令之前输入，之间没有空格。more 命令在暂停状态时立即处理子命令，并且不需要按下 Enter 键。more 命令使用下面的子命令。

- h：显示描述 more 子命令的帮助屏幕。
- v：启动 vi 编辑器，在当前行编辑当前文件。
- r^L：刷新显示。
- R：刷新显示并删除缓冲的输入。

在 more 命令的执行过程中，用户可以使用 more 自己的一系列命令动态地根据需要选择显示的部分。more 命令在显示完一屏内容之后，将停下来等待用户输入某个命令。下面列出了 more 命令在执行中用到的一些常用命令，而有关这些命令的完整内容，可以在 more 执行时按 h 键查看。这些命令的执行方法是先输入 i（行数）的值，再输入所需要的命令，否则它会以预设值来执行命令。

- i：空格：若指定 i，显示下面的 *i* 行；否则，显示下一整屏。
- i：回车：若指定 i，显示下面的 *i* 行；否则，显示下一行。
- i d：若指定 i，显示下面的 *i* 行；否则，往下显示半屏（一般为 11 行）。
- qQ：退出 more。
- =：显示当前行号。

说明

more 命令读取文件，而且每次显示一屏文本。该命令在每屏后暂停，并在屏幕底部打印单词 more。如果随后按 Enter 键，more 命令会再显示一行。如果按下 Space 键，more 命令将显示文本的另一屏。对于有些终端型号，more 命令清除屏幕，而不是滚动屏幕。可以重定向通过管道传送标准输出（如长目录列表）到 more 命令，而不是读取命名的文件。当从文件而不是管道读取的时候，命令在提示符处加%（百分符），这提供了 more 命令已经读取文件的百分比（以字符为单位，而不是行）。

应用实例如下。

（1）查看以 myfile 命名的文件。

```
#more myfile
```

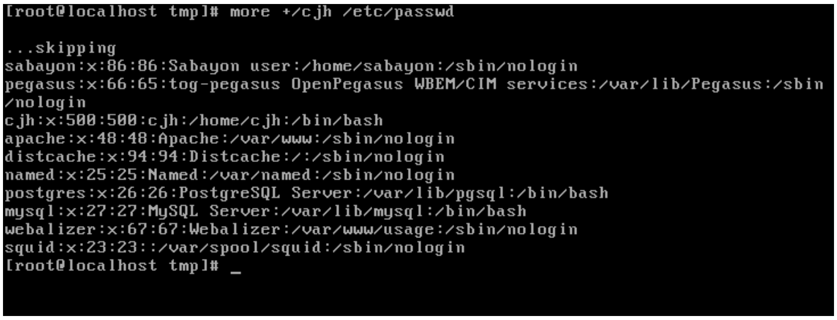
（2）从 ls 命令查看输出。

```
#ls -l | more
```

（3）逐页显示 testfile 的档案内容，如有连续两个以上空白行则以一个空白行显示。

```
#more -s  testfile
```

（4）查找/etc/passwd 文件中用户 cjh 所在行，查找过程如图 2-7 所示。



```
[root@localhost tmp]# more +/cjh /etc/passwd

...skipping
sabayon:x:86:86:Sabayon user:/home/sabayon:/sbin/nologin
pegasus:x:66:65:tog-pegasus OpenPegasus WBEM/CIM services:/var/lib/Pegasus:/sbin
/nologin
cjh:x:500:500:cjh:/home/cjh:/bin/bash
apache:x:48:48:Apache:/var/www:/sbin/nologin
distcache:x:94:94:Distcache:/:/sbin/nologin
named:x:25:25:Named:/var/named:/sbin/nologin
postgres:x:26:26:PostgreSQL Server:/var/lib/pgsql:/bin/bash
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
webalizer:x:67:67:Webalizer:/var/www/usage:/sbin/nologin
squid:x:23:23::/var/spool/squid:/sbin/nologin
[root@localhost tmp]# _
```

图 2-7 查找/etc/passwd 文件中用户 cjh 所在行

2.43 md5sum：MD5 函数值计算和检查

作用：检验 MD5 校验码。

用法：md5sum [选项] [文件]

主要选项如下。

- -b,--binary：以二进制模式读取。
- -c,--check：从文件中读取 MD5 的校验值并予以检查。
- -t,--text：以纯文本模式读取（默认）。

以下 3 个选项在进行校验时非常有用。

- --quiet：不为校验成功的文件输出 OK。
- --status：不输出任何内容，使用退出状态号显示成功。
- -w,--warn：对格式不准确的校验和行进行警告。
- --strict：验证成功时不输出任何信息。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

校验和会按照 RFC 1321 规范生成。在进行检查时，给出的输入格式应该和程序的输出样

板格式相同。默认的输出模式是输出一行校验和的校验结果，并有一个字符来表示文件类型（“*”代表二进制，“ ”代表纯文本），并同时显示每个文件的名称。

说明

MD5 算法常常被用来验证网络文件传输的完整性，防止文件被人篡改。MD5 全称是报文摘要算法（Message Digest Algorithm 5），此算法对任意长度的信息逐位进行计算，产生一个二进制、长度为 128 位（十六进制长度就是 32 位）的“指纹”（称“报文摘要”），不同的文件产生相同的报文摘要的可能性是非常小的。

应用实例如下。

（1）使用 md5sum 来产生指纹（报文摘要）。

```
#md5sum file > file.md5
```

也可以把多个文件的报文摘要输出到一个 MD5 文件中，这要使用通配符*。例如某目录下有几个.iso 文件，要把这几个.iso 文件的摘要输出到 iso.md5 文件中，命令如下：

```
#md5sum *.iso > iso.md5
```

（2）使用 MD5 报文摘要验证文件。

把下载的文件 file 和该文件的 file.md5 报文摘要文件放在同一个目录下，然后用如下命令进行验证：

```
#md5sum -c file.md5
```

如果验证成功，则会输出“正确”。

2.44 awk：模式匹配语言

作用：利用一组用户提供的命令来将一组文件和用户提供的扩展正则表达式进行比较，一次一行，然后在任何与扩展正则表达式匹配的行上执行操作。

用法：awk [options] 'script' var=value file(s)

awk [options] -f scriptfile var=value file(s)

主要选项如下。

- -F fs or --field-separator fs：指定输入文件的分隔符，fs 是一个字符串或是一个正则表达式，如-F:。
- -v var=value or --asign var=value：赋值一个用户定义变量。
- -f scripfile or --file scriptfile：从脚本文件中读取 awk 命令。
- -mf nnn and -mr nnn：对 nnn 值设置内在限制，-mf 选项限制分配给 nnn 的最大块数目；

-mr 选项限制记录的最大数目。这两个功能是 Bell 实验室版 awk 的扩展功能，在标准 awk 中不适用。

- -W compact or --compat, -W traditional or –traditional：在兼容模式下运行 awk。所以 gawk 的行为和标准的 awk 完全一样，所有的 awk 扩展都被忽略。
- -W copyleft or --copyleft, -W copyright or –copyright：打印简短的版权信息。
- -W help or --help, -W usage or –usage：打印全部 awk 选项和每个选项的简短说明。
- -W lint or –lint：打印不能向传统 UNIX 平台移植的结构的警告。
- -W lint-old or --lint-old：打印关于不能向传统 UNIX 平台移植的结构的警告。
- -W posix：打开兼容模式。但有以下限制，不识别\x、函数关键字、func、换码序列；当 fs 是一个空格时，将新行作为一个域分隔符；操作符**和**=不能代替^和^=；fflush 无效。
- -W re-interval or --re-inerval：允许间隔正则表达式的使用，参考 grep 中的 POSIX 字符类，如括号表达式[[:alpha:]]。
- -W source program-text or --source program-text：使用 program-text 作为源代码，可与-f 命令混用。

awk 脚本是由模式和操作组成的，即 pattern {action}，如$ awk '/root/' test，$ awk '$3 < 100' test。二者是可选的，如果没有模式，则 action 应用到全部记录；如果没有 action，则输出匹配全部记录。在默认情况下，每一个输入行都是一条记录，但用户可通过 RS 变量指定不同的分隔符进行分隔。

说明

awk 命令的模式搜索比 grep 命令的搜索更常用，且它允许用户在输入文本行上执行多个操作。awk 命令编程语言不需要编译，并允许用户使用变量、数字函数、字符串函数和逻辑运算符。awk 命令受到 LANG、LC_ALL、PATH 等环境变量的影响。

awk 是 Linux 下的一个命令，它对其他命令的输出、对文件的处理都十分强大。其实它更像一门编程语言，可以自定义变量，有条件语句，有循环，有数组，有正则，有函数等。它读取输出或者文件的方式是一行一行地读，根据用户给出的条件进行查找，并在找出的行中进行操作，感觉它的设计思想真的很简单，但是结合实际情况，具体操作起来就没有那么简单了。它有 3 种形式：awk，gawk，nawk，平时所说的 awk 其实就是 gawk。

1．变量

变量及其描述如表 2-4 所示。

表 2-4　变量及其描述

变　量	描　述
$n	当前记录的第 *n* 个字段，字段间由 FS 分隔
$0	完整的输入记录
ARGC	命令行参数的数目

续表

变　量	描　述
ARGIND	命令行中当前文件的位置（从 0 开始算）
ARGV	包含命令行参数的数组
CONVFMT	数字转换格式（默认值为%.6g）
ENVIRON	环境变量关联数组
ERRNO	最后一个系统错误的描述
FIELDWIDTHS	字段宽度列表（用空格键分隔）
FILENAME	当前文件名
FNR	同 NR，但相对于当前文件
FS	字段分隔符（默认是任何空格）
IGNORECASE	如果为真，则进行忽略大小写的匹配
NF	当前记录中的字段数
NR	当前记录数
OFMT	数字的输出格式（默认值是%.6g）
OFS	输出字段分隔符（默认值是一个空格）
ORS	输出记录分隔符（默认值是一个换行符）
RLENGTH	由 match 函数所匹配的字符串的长度
RS	记录分隔符（默认是一个换行符）
RSTART	由 match 函数所匹配的字符串的第一个位置
SUBSEP	数组下标分隔符（默认值是\034）

2．运算符

运算符及其描述如表 2-5 所示。

表 2-5　运算符及其描述

运 算 符	描　述
=、+=、-=、*=、/=、%=、^=、**=	赋值
?:	C 条件表达式
||	逻辑或
&&	逻辑与
~、~!	匹配正则表达式和不匹配正则表达式
<、<=、>、>=、!=、==	关系运算符
空格	连接
+、-	加、减
*、/、&	乘、除与求余
+、-、!	一元加、减和逻辑非
^、***	求幂

续表

运 算 符	描　述
++、--	增加或减少、作为前缀或后缀
$	字段引用
in	数组成员

3．awk 的正则

匹配符及其描述如表 2-6 所示。

表 2-6　匹配符及其描述

匹 配 符	描　述
\Y	匹配一个单词开头或者末尾的空字符串
\B	匹配单词内的空字符串
\<	匹配一个单词的开头的空字符串，锚定开始
\>	匹配一个单词的末尾的空字符串，锚定末尾
\W	匹配一个非字母数字组成的单词
\w	匹配一个字母数字组成的单词
\'	匹配字符串末尾的一个空字符串
\‘	匹配字符串开头的一个空字符串

4．字符串函数

字符串函数及其描述如表 2-7 所示。

表 2-7　字符串函数及其描述

函 数 名	描　述
sub	匹配记录中最大、最靠左边的子字符串的正则表达式，并用替换字符串替换这些字符串。如果没有指定目标字符串，就默认使用整个记录。替换只发生在第一次匹配的时候
gsub	在整个文档中进行匹配
index	返回子字符串第一次被匹配的位置，偏移量从位置 1 开始
substr	返回从位置 1 开始的子字符串，如果指定长度超过实际长度，则返回整个字符串
split	可按给定的分隔符把字符串分隔为一个数组。如果分隔符没有提供，则按当前 FS 值进行分隔
length	返回记录的字符数
match	返回在字符串中正则表达式位置的索引，如果找不到指定的正则表达式则返回 0。match 函数会设置内建变量 RSTART 为字符串中子字符串的开始位置，RLENGTH 为到子字符串末尾的字符个数。substr 可利于这些变量来截取字符串
toupper 和 tolower	可用于字符串大小间的转换，该功能只在 gawk 中有效

5．数学函数

数学函数及其返回值如表 2-8 所示。

表 2-8　数学函数及其返回值

函 数 名	返 回 值
atan2(*x*,*y*)	*y*,*x* 范围内的余切
cos(*x*)	余弦函数
exp(*x*)	求幂
int(*x*)	取整
log(*x*)	自然对数
rand()	随机数
sin(*x*)	正弦
sqrt(*x*)	平方根
srand(*x*)	*x* 是 rand()函数的种子
int(*x*)	取整，过程没有舍入
rand()	产生一个大于等于 0 而小于 1 的随机数

应用实例如下。

（1）显示文件 hello.c 中的行号和第三字段。

```
#awk '{printf"%03d%s\n",NR,$1}' hello.c
001/*
002#include
003int
004{
005printf("Hello,
006return
007}
```

（2）显示长于 7 个字符的文件的行。

```
# awk  'length  >7'  hello.c
/* hello.c - Canonical "Hello, World!" program */
#include <stdio.h>
int main(void)
       printf("Hello, Linux programming world!\n");
       return 0;
```

（3）以相反顺序打印前两个字段。

```
awk  'length  >7'  hello.c
/* hello.c - Canonical "Hello, World!" program */
#include <stdio.h>
int main(void)
       printf("Hello, Linux programming world!\n");
```

```
        return 0;
[root@localhost tmp]# awk '{ print $2, $1 }' hello.c
hello.c /*
<stdio.h> #include
main(void) int
 {
Linux printf("Hello,
0; return
 }
```

2.45 wc：输出文件中的行数、单词数、字节数

作用：对每个文件输出行、单词和字节统计数，如果指定了多于一个文件，则还有一个行数的总计。如果指定的文件是“-”，则读取标准输入。

用法：wc [选项]...[文件]

主要选项如下。

- -c, --bytes, --chars：输出字节统计数。
- -l, --lines：输出换行符统计数。
- -L, --max-line-length：输出最长的行的长度。
- -w, --words：输出单词统计数。

应用实例如下。

显示文件 hello.c 中的单词数。

```
# wc -w hello.c
20 hello.c
```

2.46 comm：比较排序文件

作用：逐行比较两个已排序的文件。

用法：comm [选项]...文件 1 文件 2

主要选项如下。

如果不附带选项，程序会生成三列输出。第一列包含文件 1 特有的行，第二列包含文件 2 特有的行，而第三列包含两个文件共有的行。

- -1：不输出文件 1 特有的行。
- -2：不输出文件 2 特有的行。

– -3：不输出两个文件共有的行。
– --check-order：检查输入是否被正确排序，即使所有输入行均成对。
– --nocheck-order：不检查输入是否被正确排序。
– --output-delimiter=STR：依照 STR 分列。
– --help：显示此帮助信息并退出。
– --version：显示版本信息并退出。

应用实例如下。

比较排序 FILE1 和 FILE2，共有部分不显示。

```
#comm -3 FILE1 FILE2
SSSS
222
```

2.47 join：将两个文件中指定栏位内容相同的行连接起来

作用：找出两个文件中指定栏位内容相同的行并加以合并，再输出到标准输出设备。

用法：join [选项]文件 1 文件 2

主要选项如下。

针对每一对具有相同内容的输入行，整合为一行写到标准输出，默认的内容连接区块是由第一个空白符代表的分界符号。当文件 1 或文件 2 都被指定为“-”时，程序将从标准输入读取数据。

– -a 文件编号：文件编号的值可以是 1 或 2，分别对应文件 1 和文件 2。此选项用于根据指定文件编号输出不成对的行目。
– -e 字符：将缺失的输入区块替换为指定字符。
– -i,--ignore-case：比较时忽略大小写。
– -j 域：等于“-1 域-2 域”。
– -o 格式：按照指定格式构造输出行。
– -t 字符：使用指定字符作为输入和输出的分隔符。
– -v 文件编号：类似-a 文件编号，但禁止组合输出行。
– -1 域：在文件 1 的此域组合。
– -2 域：在文件 2 的此域组合。
– --check-order：检查输入行是否正确排序，即使所有输入行均是成对的。
– --nocheck-order：不检查输入是否正确排序。
– --header：将首行视作域的头部，直接输出而不对其进行匹配。
– --help：显示此帮助信息并退出。
– --version：显示版本信息并退出。

除非使用了“-t 字符串”选项，否则前导空格分隔的域将被忽略。如果指定了字符串，则

使用指定字符串分隔任意的域并从 1 开始计数的域编号。可以指定的格式是由一个或多个逗号或空格所分隔的描述，其形式为“文件编号.域”或者“0”。默认的格式输出合并后的域、文件 1 和文件 2 剩下的域，均由该指定字符串分隔。

重要提示：文件 1 和文件 2 必须在合并域中排序。

例如，如果“join”后没有选项，则使用“sort -k 1b,1”。

> **注意**
>
> 所进行的比较遵从“LC_COLLATE”所指定的规则。如果输入没有被排序并导致某些行无法合并，将会显示警告信息。

应用实例如下。

示例代码如下。

```
# join -1 4 -2 3 -o 1.1 -o 2.1 -o 1.6 -t":" /etc/passwd /etc/group
root:root:/root
bin:bin:/bin
daemon:daemon:/sbin
adm:adm:/var/adm
lp:lp:/var/spool/lpd
nobody:nobody:/
vcsa:vcsa:/dev
rpm:rpm:/var/lib/rpm
nscd:nscd:/
ident:ident:/home/ident
netdump:netdump:/var/crash
sshd:sshd:/var/empty/sshd
rpc:rpc:/
...
```

在实践中使用 join 工具最明显的方法是从/etc/passwd 中提取用户名和对应的主目录项，并从/etc/group 文件中提取组名。组名在/etc/passwd 文件中是以数字的格式出现在第 4 个域中的。类似地，它们在/etc/group 文件中是在第 3 个域中出现的。

2.48 fmt：编排文本文件

作用：从指定的文件里读取内容，将其按照指定格式重新编排后，输出到标准输出设备。若指定的文件名为“-”，则 fmt 命令会从标准输入设备读取数据。

用法：fmt [-宽度] [选项]... [文件]...

重新格式化文件中的每个段落并输出到标准输出。

选项“-宽度”是“--width=数字”的缩写。

主要选项如下。

长选项必须使用的参数对于短选项也是必须使用的。

- -c --crown-margin：保持前两行的缩进。
- -p,--prefix=字符串：只对以指定字符串开头的行重新格式化，将前缀重新附着到被重新格式化的行上。
- -s,--split-only：分割过长的行，但不自动补足。
- -t,--tagged-paragraph：缩进首行以使其不同于第二行。
- -u,--uniform-spacing：每两个单词间保留一个空格，每句之后保留两个空格。
- -w,--width=宽度：最大行宽（默认为 75 列宽度）。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

应用实例如下。

重新排版 hello.c 文件。

```
# fmt hello.c
/* hello.c - Canonical "Hello, World!" program */ #include <stdio.h>
int main(void) {
        printf("Hello, Linux programming world!\n"); return 0;
}
```

2.49 tr：转换字符

作用：从标准输入设备读取数据，经过字符串转义后，输出到标准输出设备。

用法：tr [选项]... SET1 [SET2]

主要选项如下。

从标准输入中替换、缩减和/或删除字符，并将结果写到标准输出。

- -c,-C,--complement：首先补足 SET1。
- -d,--delete：删除匹配 SET1 的内容，并不作替换。
- -s,--squeeze-repeats：如果匹配于 SET1 的字符在输入序列中存在连续的重复，在替换时会被统一缩为一个字符的长度。
- -t,--truncate-set1：先将 SET1 的长度截为和 SET2 相等。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

SET 是一组字符串，一般可按照字面含义理解。解析序列如下。

- \NNN：八进制值为 NNN 的字符（1~3 个数位）。

- \\：反斜杠。
- \a：终端鸣响。
- \b：退格。
- \f：换页。
- \n：换行。
- \r：回车。
- \t：水平制表符。
- \v：垂直制表符。
- 字符 1-字符 2：从字符 1 到字符 2 的升序递增过程中经历的所有字符。
- [字符*]：在 SET2 中适用，指定字符会被连续复制直到吻合设置 1 的长度。
- [字符*次数]：对字符执行指定次数的复制，若次数以 0 开头则被视为八进制数。
- [:alnum:]：所有的字母和数字。
- [:alpha:]：所有的字母。
- [:blank:]：所有呈水平排列的空白字符。
- [:cntrl:]：所有的控制字符。
- [:digit:]：所有的数字。
- [:graph:]：所有的可打印字符，不包括空格。
- [:lower:]：所有的小写字母。
- [:print:]：所有的可打印字符，包括空格。
- [:punct:]：所有的标点字符。
- [:space:]：所有呈水平或垂直排列的空白字符。
- [:upper:]：所有的大写字母。
- [:xdigit:]：所有的十六进制数。
- [=字符=]：所有和指定字符相等的字符。

仅在 SET1 和 SET2 都给出，同时没有-d 选项的时候才会进行替换。仅在替换时才可能用到-t 选项。如果需要，SET2 将被通过在末尾添加原来的末字符的方式补充到同 SET1 等长，SET2 中多余的字符将被省略。只有[:lower:]和[:upper:]以升序展开字符；在用于替换时的 SET2 中以成对表示大小写转换。-s 作用于 SET1，既不替换也不删除，否则在替换或展开后使用 SET2 缩减。

应用实例如下。

将 hello.c 中的文字大小写进行转换。

```
# cat hello.c | tr a-x A-X
/* HELLO.C - CANONICAL "HELLO, WORLD!" PROGRAM */
#INCLUDE <STDIO.H>
INT MAIN(VOID)
{
      PRINTF("HELLO, LINUX PROGRAMMING WORLD!\N");
      RETURN 0;
}
```

2.50 col：过滤控制字符

作用：在许多 UNIX 说明文件里，都有 RLF 控制字符。当我们运用 Shell 特殊字符“>”和“>>”把说明文件的内容输出成纯文本文件时，控制字符会变成乱码，col 命令则能有效滤除这些控制字符。

用法：col [-bfpx] [-l nline]

主要选项如下。

- -b：过滤掉所有的控制字符，包括 RLF 和 HRLF。
- -f：滤除 RLF 字符，但允许将 HRLF 字符呈现出来。
- -x：以多个空格字符来表示跳格字符。
- -l nline：预设的内存缓冲区有 128 列，可以自行指定缓冲区的大小。

应用实例如下。

过滤 cat 手册页的 RLF 和 HRLF 字符并且重定向到文件 cat.man.txt。

```
# man cat | col -b > cat.man.txt
```

2.51 colrm：删除指定的行

作用：从标准输入设备读取数据，转而输出到标准输出设备。如果不加任何参数，则该命令不会过滤任何一行。

用法：colrm [起始列 [终止列]]

主要选项如下。

- -V, --version：输出版本信息并退出。
- -h, --help：显示此帮助并退出。

应用实例如下。

删除 hello.c 文件第二行。

```
# colrm 2 < hello.c
```

2.52 fold：限制文件列宽

作用：从指定的文件里读取内容，将超过限定列宽的列加入增列字符后，输出到标准输出设备。若不指定任何文件名称，而所给予的文件名为“-”，则 fold 命令会从标准输入设备读取数据。

用法：fold [选项]... [文件]...

主要选项如下。

– -b—bytes：以 Byte 为单位计算列宽，而非采用行数编号为单位。

– -s—spaces：以空格字符作为换列点。

– -w<每列行数>--width<每列行数>：使用指定的列宽代替默认的 80。

应用实例如下。

调整 hello.c 最多宽度为 25 个字符显示。

```
# fold -w 25 hello.c
```

2.53 iconv：转换给定文件的编码

作用：转换给定文件的编码。

用法：iconv [选项...] [文件...]

主要选项如下。

输入/输出格式规范如下。

– -f, --from-code=名称：原始文本编码。

– -t, --to-code=名称：输出编码。

信息如下。

-l, --list：列举所有已知的字符集。

输出控制如下。

– -c：从输出中忽略无效的字符。

– -o, --output=FILE：输出文件。

– -s, --silent：关闭警告。

– verbose：打印进度信息。

– -?, --help：给出该系统求助列表。

– usage：给出简要的用法信息。

– -V, --version：打印程序版本号。

应用实例如下。

显示目前系统支持的字符集。

```
# iconv -l|more
437//
500//
500V1//
850//
…
```

2.54 dc：任意精度的计算器

作用：dc 命令提供了一种交互式桌面计算器来完成任意精度的整数计算。

用法：dc [选项] [file...]

主要选项如下。

- -e, --expression=EXPR：计算表达式。
- -f, --file=FILE：计算目录文件。

子命令如下。

- c：清洁堆栈。dc 命令弹出堆栈中的所有值。
- d：复制栈顶值。
- f：显示堆栈上的所有值。
- i：弹出栈顶值并将其作为进一步输入的基数。
- I：将输入基数压入栈顶。
- k：弹出栈顶值并将其作为非负比例因子。
- o：将栈顶值弹出并将其作为进一步输出的基数。
- O：将输出基数压入栈顶。
- P：显示栈顶值。顶部值不会发生改变。
- q：退出程序。如果 dc 命令正在运行一个字符串，它将弹出递归级别为 2。

运算类型如下。

+，-，/，*，%，^：对堆栈顶部的两个数进行加（+）、减（-）、乘（*）、除（/）、求余（%）或取幂（^）。

应用实例如下。

使用 dc 命令进行计算。

```
# dc
7
8
*
p
56
q
```

2.55 expr：求表达式变量的值

作用：expr 命令读入表达式 Expression 的参数，计算它的值，然后将结果写入到标准输出。

用法：expr 表达式

　　expr 选项

主要选项如下。

- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

将表达式的值列印到标准输出，分隔符下面的空行可提升算式优先级。

可用的表达式如下。

- ARG1|ARG2：若 ARG1 的值不为 0 或者为空，则返回 ARG1，否则返回 ARG2。
- ARG1&ARG2：若两边的值都不为 0 或为空，则返回 ARG1，否则返回 0。
- ARG1<ARG2：ARG1 小于 ARG2。
- ARG1<=ARG2：ARG1 小于或等于 ARG2。
- ARG1=ARG2：ARG1 等于 ARG2。
- ARG1!=ARG2：ARG1 不等于 ARG2。
- ARG1>=ARG2：ARG1 大于或等于 ARG2。
- ARG1>ARG2：ARG1 大于 ARG2。
- ARG1+ARG2：计算 ARG1 与 ARG2 相加之和。
- ARG1-ARG2：计算 ARG1 与 ARG2 相减之差。
- ARG1*ARG2：计算 ARG1 与 ARG2 相乘之积。
- ARG1/ARG2：计算 ARG1 与 ARG2 相除之商。
- ARG1%ARG2：计算 ARG1 与 ARG2 相除之余数。
- 字符串:表达式：定位字符串中匹配表达式的模式。
- match 字符串 表达式：等于“字符串:表达式”。
- substr 字符串 偏移量 长度：替换字符串的子串，偏移的数值从 1 起计。
- index 字符串 字符：在字符串中发现字符的地方建立下标，或者标 0。
- length 字符串：字符串的长度。
- +TOKEN：将 TOKEN 解释为串，不管它是一个关键字或者一个操作。
- (EXPRESSION)：EXPRESSION 的数值。

请注意有许多运算操作符都可能需要由 Shell 先实施转义。

如果参与运算的 ARG 自变量都是数字，比较符就会被视作数学符号，否则就是多义的。

模式匹配会返回“\”和“\”之间被匹配的子字符串或空（null）；如果未使用“\”和“\”，则会返回匹配字符数量或 0。

若表达式的值既不是空也不是 0，则退出状态值为 0；若表达式的值为空或为 0，则退出状态值为 1；如果表达式的句法无效，则在出错时返回退出状态值 3。

应用实例如下。

（1）整数运算。

```
expr 14 % 9
5
```

（2）返回“fd”字符串的任何字符在“abcdef”中出现的第一个位置。

```
# expr index abcdef fd
4
```

（3）返回“Goodnight Ladies”字符串中从位置 11 开始的 6 个字符长的字符串。

```
# expr substr "Goodnight Ladies" 11 6
Ladies
```

（4）计算字串长度。

```
# expr length thisisatest
11
```

2.56 strings：显示文件中的可打印字符

作用：显示每个指定的文件中包含的所有有 4 个（或用选项指定的数字）以上连续可打印字符的字符串，在之后紧跟着一个不可打印的字符。

用法：strings [选项] file

主要选项如下。

- -a --all：不仅扫描目标文件初始化和装载段，而是扫描整个文件。
- --print-file-name–f：在显示字符串之前先显示文件名。
- -bytes=min-len：打印至少 min-len 字符长的字符串，默认的是 4。
- --radix={o,x,d}：在字符串前面显示其在文件中的偏移量。选项中的单个字符指定偏移量的进制是八进制、十六进制还是十进制。
- --target=bfdname：指定一种目标代码格式来代替系统的默认格式。

应用实例如下。

显示一个文件中大于等于 4 个字符的字符串。

```
strings scsrun.sh
#!/bin/sh
skip=14
```

2.57 xargs：从标准输入读入参数

作用：从标准输入读入参数，参数用空格（可以用双引号、单引号反斜杠转义）或回车符隔开，然后一次或多次执行命令。xargs 是一条 UNIX 和类 UNIX 操作系统的常用命令，它的作用是将参数列表转换成小块分段传递给其他命令，以避免参数列表过长的问题。

用法：xargs[选项]

主要选项如下。

- --null,-0：输入的文件名以 null 符结尾，而不是空格，引号和反斜杠并不做特殊处理。
- --eof[=eof-str],-e[eof-str]：文件尾字符串设置成 eof-str。如果文件尾字符串出现在输入中的某行，则余下的行将被忽略。
- --replace[=replace-str],-i[replace-str]：把 initial-arguments 里的所有 replace-str 替换为从标准输入里读入的名称。
- --max-lines[=max-lines],-l[max-lines]：每个命令行最多可以有 max-lines 行非空格输入，max-lines 默认是 1。
- --max-args=max-args,-nmax-args：每个命令行最多可以有 max-args 个参数。
- --interactive,-p：提示用户是否运行每个命令行，然后从终端读入一行。只有当此行以‘y’、‘Y’开始时才会运行此命令行。
- --no-run-if-empty,-r：如果标准输入不包含任何非空格，将不运行命令。
- --max-chars=max-chars,-smax-chars：每个命令行最多可以有 max-chars 个字符。
- --arg-file=file：从指定文件读取参数。
- -i：用{}代替传递的数据。
- -I string：用 string 代替传递的数据。
- -n[数字]：设置每次传递几行数据。
- -t：显示执行详情。
- -p：交互模式。
- -P n：允许的最大线程数量为 n。
- -s[大小]：设置传递参数的最大字节数（小于 131 072 字节）。
- -x：大于-s 设置的最大长度结束 xargs 命令执行。

xargs 退出可能有如下几种状态。

- 0：如果成功。
- 123：如果任何一个被调用的命令 command 退出状态为 1-125。
- 124：如果命令 command 退出状态为 255。
- 125：如果命令 command 被信号终止。
- 126：如果不能执行命令 command。
- 127：如果命令 command 没有找到。
- 1：如果发生其他错误。

应用实例如下。

下面通过一些例子说明 xargs 命令是如何同 find 命令一起使用的。

（1）查找系统中的每个普通文件，然后使用 xargs 命令来测试它们分别属于哪类文件。

```
#find . -type f -print | xargs file
./.kde/Autostart/Autorun.desktop: UTF-8 Unicode English text
./.kde/Autostart/.directory:      ISO-8859 text\
...
```

（2）在整个系统中查找内存信息转储文件（core dump），然后把结果保存到/tmp/core.log文件中。

```
$ find / -name "core" -print | xargs echo "" >/tmp/core.log
```

（3）改用 xargs 命令。

执行下面的命令：

```
#rm 'find /path -type f'
```

如果 path 目录下文件过多，就会因为“参数列表过长”而报错无法执行。但改用 xargs 命令以后，问题即可解决。

```
#find /path -type f -print0 | xargs -0 rm
```

说明

xargs 的作用等同于大多数 UNIX Shell 中的反引号，但更加灵活易用，并且可以正确处理输入中有空格等特殊字符的情况，对于经常产生大量输出的命令（如 find、locate 和 grep）来说非常有用。在使用 find 命令的-exec 选项处理匹配到的文件时，find 命令将所有匹配到的文件一起传递给 exec 执行。不幸的是，有些系统对能够传递给 exec 的命令长度有限制，这样在 find 命令运行几分钟之后，就会出现溢出错误。错误信息通常是“参数列太长”、“参数列溢出”。这就是 xargs 命令的用处所在，特别是与 find 命令一起使用。find 命令把匹配到的文件传递给 xargs 命令，而 xargs 命令每次只获取一部分文件而不是全部，不像-exec 选项那样。这样它可以先处理最先获取的一部分文件，然后是下一批，并如此继续下去。在有些系统中，使用-exec 选项会为处理每一个匹配到的文件而发起一个相应的进程，并非将匹配到的文件全部作为参数一次执行。这样在有些情况下就会出现进程过多、系统性能下降的问题，因而效率不高，而使用 xargs 命令只有一个进程。另外，在使用 xargs 命令时，究竟是一次获取所有的参数，还是分批取得参数，以及每次获取参数的数目，都会根据该命令的选项及系统内核中相应的可调参数来确定。

2.58 sum：计算文件的校验和，以及文件占用的块数

作用：sum 命令显示文件的校验和及文件块数。

用法：sum [选项]...[文件]

为每个指定的文件输出校验和及块计数。

主要选项如下。

- -r：使用 BSD 加和算法，以 1KB 为一个块。
- -s,--sysv：使用 SystemV 加和算法，以 51 双字节为一个块。

- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

如果没有指定文件，或者文件为“-”，则从标准输入读取。

应用实例如下。

显示 hello.o 和 hello.s 文件中的校验和，以及 1024 字节块数目，请输入如下命令：

```
# sum hello.o hello.s
31784     1 hello.o
48706     1 hello.s
```

显示的第一个字段是校验和，第二个字段是 1024 字节块数目，第三个字段是文件名。

2.59 setfacl：设定文件访问控制列表

作用：设定文件访问控制列表。

用法：setfacl [-bkndRLP] { -m|-M|-x|-X ... } file...

主要选项如下。

- -m,--modify=acl：更改文件的访问控制列表。
- -M,--modify-file=file：从文件读取访问控制列表条目并更改。
- -x,--remove=acl：根据文件中的访问控制列表移除条目。
- -X,--remove-file=file：从文件读取访问控制列表条目并删除。
- -b,--remove-all：删除所有扩展访问控制列表条目。
- -k,--remove-default：移除默认访问控制列表。
- --set=acl：设定替换当前的文件访问控制列表。
- --set-file=file：从文件中读取访问控制列表条目设定。
- --mask：重新计算有效权限掩码。
- -n,--no-mask：不重新计算有效权限掩码。
- -d,--default：应用到默认访问控制列表的操作。
- -R,--recursive：递归操作子目录。
- -L,--logical：依照系统逻辑，跟随符号链接。
- -P,--physical：依照自然逻辑，不跟随符号链接。
- --restore=file：恢复访问控制列表，和“getfacl-R”作用相反。
- --test：测试模式，并不真正修改访问控制列表属性。
- -v,--version：显示版本信息并退出。
- -h,--help：显示帮助信息。

应用实例如下。

（1）修改文件的 acl 权限，添加一个用户权限。

```
# setfacl -m u:zhangy:rw- test
```

（2）添加一个组。

```
#setfacl -m g:zhangying:r-w test
```

2.60 getfacl：获取文件访问控制列表

作用：获取文件访问控制列表。

用法：getfacl[-aceEsRLPtpndvh]文件

主要选项如下。

- -a,--access：仅显示文件访问控制列表。
- -d,--default：仅显示默认的访问控制列表。
- -c,--omit-header：不显示注释表头。
- -e,--all-effective：显示所有的有效权限。
- -E,--no-effective：显示无效权限。
- -s,--skip-base：跳过只有基条目（Baseentries）的文件。
- -R,--recursive：递归显示子目录。
- -L,--logical：逻辑遍历（跟随符号链接）。
- -P,--physical：物理遍历（不跟随符号链接）。
- -t,--tabular：使用制表符分隔的输出格式。
- -n,--numeric：显示数字的用户/组标识。
- -p,--absolute-names：不去除路径前的“/”符号。
- -v,--version：显示版本并退出。
- -h,--help：显示帮助信息。

应用实例如下。

（1）查看 acl 权限列表。

```
# getfacl test
# file: test
# owner: root
# group: root
user::r-x
user:tank:rwx                #effective:---
group::r-x                  #effective:---
mask::---
other::---
```

（2）查看 acl 权限，省略头信息。

```
# getfacl -c test
user::r-x
```

```
user:tank:rwx                    #effective:---
group::r-x                       #effective:---
mask::---
other::---
```

2.61 chacl：更改文件或目录的访问控制列表

作用：更改文件或目录的访问控制列表。

用法：chacl acl filename1 filename2

主要选项如下。

- -b：表明这里有两个 acl 需要修改，前一个 acl 是文件的 acl，后一个是目录的默认 acl。
- -d：设定目录的默认 acl，这个选项是比较有用的。如果指定了目录的默认 acl，在这个目录下新建的文件或目录都会继承目录的 acl。
- -R：只删除文件的 acl。
- -D：只删除目录的默认 acl，是-d 的反向操作。
- -B：删除文件和目录默认的 acl，是-b 的反向操作。
- -r：递归地修改文件和目录的 acl 权限。
- -l：列出文件和目录的 acl 权限。

> **说明**
>
> chacl 是用来更改文件或目录的访问控制列表的命令。其和 chmod 有异曲同工之妙，但是比 chmod 更为强大、更为精细。

chmod 只能把权限分为 3 种：用户、组、其他人。通过 chmod 可以指定文件属主的权限、文件所在组的用户的权限，以及其他人的权限。

如果有这样的需求，通过 chmod 能搞定吗？如果 A 用户的文件只想给 B 看，通过 chmod 能不能搞定？当然读者可能认为把 A、B 放到一个组里面就可以了，但是这样就会限制了其他用户不能加入这个组。通过 chacl 可以轻松地解决这个问题。

chacl 命令的格式如下：

```
chacl acl filename1 filename2
```

acl 的英文是 the access control list，直译为存取控制列表。chacl 就是一个改变文件存取控制列表的命令。

acl 是一个可以被 acl_from_text 程序分析出各用户权限的字符串，该字符串用逗号分隔成多个片段，每个片段的形式都如 tag:name:perm。

tag 可以是下面形式的一种。

```
"user"(or"u")
```

表示这是一个用户的 ACL 条目。

```
"group"(or"g")
```

表示这是一个用户组的 ACL 条目。

```
"other"(or"o")
```

表示这是其他的 ACL 条目，即没有在 ACL 指定的用户和组的 ACL 条目。

```
"mask"(or"m")
```

表示这是一个掩码的 ACL 条目。在指定其他非用户属主的 ACL 权限时，这个掩码条目必须被指定，否则执行 chacl 命令会出错。

name 可以是用户名或组名。如果不指定，那么默认是给文件或目录的属主或用户组指定 ACL 权限。当然，name 也可以是用户的 UID 或者组的 GID。

perm 是指该用户或组所具有的权限，它是由“rwx”组成的一个字符串。如果要使该用户或组不具有某种权限，简单地以“-”代替指定的字母就可以了。例如“r-x”表示只具有读、执行权限。

应用实例如下。

```
# chacl u::r-x,g::r-x,o::---,u:tank:rwx,m::--- test
# getfacl test   #查看文件的 acl 权限
# file: test
# owner: root
# group: root
user::r-x
user:tank:rwx                 #effective:---
group::r-x                    #effective:---
mask::---
other::---
```

03 Linux 磁盘管理命令

3.1 df：显示报告文件系统磁盘使用信息

作用：查看磁盘信息，显示磁盘的文件系统与使用情形。

用法：df [选项]... [文件]...

主要选项如下。

- -a 或--all：列出包括 BLOCK 为 0 的文件系统。
- --block-size=<区块大小>：以指定的区块大小来显示区块数目。
- -h 或--human-readable：以可读性较高的方式来显示信息。
- -H 或--si：与-h 参数相同，但在计算时以 1000B 为换算单位而非 1024B。
- -i 或--inodes：显示 inode 的信息。
- -k 或--kilobytes：指定区块大小为 1024B。
- -l 或--local：仅显示本地端的文件系统。
- -m 或--megabytes：指定区块大小为 1 048 576B。
- --no-sync：在取得磁盘使用信息前，不要执行 sync 指令，此为预设值。
- -P 或--portability：使用 POSIX 的输出用法。
- --sync：在取得磁盘使用信息前，先执行 sync 指令。
- -t<文件系统类型>或--type=<文件系统类型>：仅显示指定文件系统类型的磁盘信息。
- -T 或--print-type：显示文件系统的类型。
- -x<文件系统类型>或--exclude-type=<文件系统类型>：不要显示指定文件系统类型的磁盘信息。
- --help：显示帮助信息。
- --version：显示版本信息。
- [文件]：指定磁盘设备。

应用实例如下。

（1）显示所有文件系统对 I-node 节点和磁盘块的使用情况，这时需要一个参数-i。

```
#  df -i
文件系统                  Inode (I)已用 (I)可用 (I)已用% 挂载点
```

```
/dev/mapper/VolGroup00-LogVol00
                 1795200  143723 1651477    9% /
/dev/sda1           26104      43   26061    1% /boot
tmpfs               48690       1   48689    1% /dev/shm
none                48668       4   48664    1% /var/lib/xenstored
```

从上面这些信息可以看出每个文件系统中有多少可用的 I-node、其中有多少已被使用、还剩余多少、它们所占的比例等整个硬盘的使用情况。因此，系统管理员需要了解这些情况。

有时会发现这样的情况：某些硬盘的容量超过了 100%。这是因为 Linux 系统为超级用户保留了 10%的空间由它单独支配。也就是说，对于超级用户而言，它所见到的硬盘容量将是 110%。这样的安排对于系统管理而言是有好处的，当硬盘被使用的容量接近 100%时，系统管理员还可以正常工作。df 工具程序被广泛地用来生成文件系统的使用统计数据。它能显示系统中所有文件系统的信息，包括它们的总容量、可用的空闲空间、目前的安装点等。

（2）以 MB 为单位显示/home 目录的文件系统使用情况。

```
## df –m /home/
文件系统              1M-块      已用      可用 已用% 挂载点
/dev/mapper/VolGroup00-LogVol00
                       6789      3895      2544  61% /
```

3.2 du：显示目录或者文件所占的磁盘空间

作用：du 命令能显示关于硬盘使用情况的信息，它能显示某个目录下的所有文件和子目录占用硬盘空间的大小。

用法：du [选项]...[文件]...

主要选项如下。

- a 或-all：显示目录中个别文件的大小。
- -b 或-bytes：显示目录或文件大小时，以字节为单位。
- -c 或--total：除了显示个别目录或文件的大小外，同时显示所有目录或文件的大小总和。
- -D 或--dereference-args：显示指定符号链接的源文件大小。
- -h 或--human-readable：以 KB、MB、GB 为单位，提高信息的可读性。
- -H 或--si：与-h 参数相同，但是 KB、MB、GB 以 1000 为换算单位。
- -k 或--kilobytes：以 1024B 为单位。
- -l 或--count-links：重复计算硬件链接的文件。
- -L<符号链接>或--dereference<符号链接>：显示选项中所指定符号链接的源文件大小。
- -m 或--megabytes：以 MB 为单位。
- -s 或--summarize：仅显示总计。
- -S 或--separate-dirs：显示个别目录的大小时，并不包含其子目录的大小。

- -x 或--one-file-xystem：以一开始处理时的文件系统为准，若遇到其他不同的文件系统目录则略过。
- -X<文件>或--exclude-from=<文件>：在<文件>中指定目录或文件。
- --exclude=<目录或文件>：略过指定的目录或文件。
- --max-depth=<目录层数>：超过指定层数的目录后，予以忽略。
- -0, --null：结束于 0 字节的输出，而不是换行符。
- --help：显示帮助。
- --version：显示版本信息。

应用实例如下。

（1）查看 tmp 目录使用情况。

```
#  du /tmp
4       /tmp/.X11-unix
4       /tmp/VMwareDnD/23c61099
12      /tmp/VMwareDnD/66ae4cf6
20      /tmp/VMwareDnD
4       /tmp/.font-unix
4       /tmp/.ICE-unix
8       /tmp/gconfd-root
8       /tmp/orbit-root
12      /tmp/tmp
68      /tmp
```

通过上面的命令可以看到，每个命令占用的硬盘空间大小以块为单位，列在每行的最前面，后面跟着目录名称。

（2）显示/usr 目录占用的空间，以 GB 为单位。

```
 #du -s --si /usr
2.8G   /usr
```

（3）显示/tmp 目录下的子目录占用的空间。

```
du /tmp --max-depth=1
4       /tmp/.X11-unix
20      /tmp/VMwareDnD
4       /tmp/.font-unix
4       /tmp/.ICE-unix
8       /tmp/gconfd-root
8       /tmp/orbit-root
12      /tmp/tmp
68      /tmp
```

3.3 dd：磁盘操作

作用：从标准输入或文件读取数据，依指定的用法来转换数据，再输出到文件、设备或标准输出。

用法：dd [操作符]...

dd 选项

主要选项如下。

- bs=<字节数>：设置输入/输出的字节数。
- cbs=<字节数>：设置转换的字节数。
- cout=<区块数>：指定读取的区块数。
- ibs=<字节数>：设置每次读取的字节数。
- if=<文件>：指定文件作为读取的对象。
- obs=<字节数>：设置每次输出的字节数。
- of=<文件>：将结果输出到指定的文件。
- seek=<区块数>：忽略指定的区块数，不进行输出。
- skip=<区块数>：忽略指定的区块数，不进行读取。
- --help：显示帮助信息。
- --version：显示版本信息。
- conv=<关键字>：转换方式。转换方式包括如下几种。
  - ascii：将 EBCDIC 转换成 ASCII。
  - ebcdic：将 ASCII 转换成 EBCDIC。
  - ibm：将 ASCII 转换成 alternative EBCDIC。
  - block：每一行输入，无论长短，输出都是 cbs 字节。
  - unblock：用 lq 换行 rq 替换每个输入块（cbs 字节大小）末尾的空格。
  - lcase：将大写字母转换成小写字母。
  - ucase：将小写字母转换成大写字母。
  - swab：交换每对输入字节。如果读入的字节数是奇数，则最后一个字节只是简单地复制到输出（因为没有能跟它交换的字节了）。
  - noerror：发生读错误时，继续进行。
  - notrunc：不截断输出文件。
  - syn：c 用 0 填充到每个输入块的末尾，使其大小为 ibs 字节。

说明

dd 是 Linux 的一个非常有用的命令，其作用是用指定大小的块复制一个文件，并在复制的同时进行指定的转换。

应用实例如下。

（1）软盘互相复制。

要把一张软盘的内容复制到另一张软盘上，利用/tmp 作为临时存储区。把源盘插入驱动器中，输入下述命令：

```
$ dd if =/dev/fd0 of = /tmp/tmpfile
```

复制完成后，将源盘从驱动器中取出，把目标盘插入，输入如下命令：

```
$ dd if = /tmp/tmpfile of =/dev/fd0
```

复制完成后，应该将临时文件删除。

```
$ rm /tmp/tmpfile
```

（2）把 net.i 文件写入软盘中，并设定读/写缓冲区的数目（注意：软盘中的内容会被完全覆盖掉）。

```
$ dd if = net.i of = /dev/fd0 bs = 16384
```

（3）将文件 sfile 复制到文件 dfile 中。

```
$ dd if=sfile of=dfile
```

（4）整盘数据备份与恢复。

① 整盘数据备份。

将本地的/dev/hdx 整盘备份到/dev/hdy。

```
dd if=/dev/hdx of=/dev/hdy
```

将/dev/hdx 全盘数据备份到指定路径的 image 文件。

```
dd if=/dev/hdx of=/path/to/image
```

备份/dev/hdx 全盘数据，并利用 gzip 工具进行压缩，保存到指定路径。

```
dd if=/dev/hdx | gzip /path/to/image.gz
```

② 整盘数据恢复。

将备份文件恢复到指定盘。

```
dd if=/path/to/image of=/dev/hdx
```

将压缩的备份文件恢复到指定盘。

```
gzip -dc /path/to/image.gz | dd of=/dev/hdx
```

（5）备份与恢复 MBR 记录。

① 备份 MBR 记录。

备份磁盘开始的 512B 的 MBR 信息到指定文件。

```
dd if=/dev/hdx of=/path/to/image
count=1 bs=512
```

② 恢复 MBR 记录。

将备份的 MBR 信息写到磁盘开始部分。

```
dd if=/path/to/image of=/dev/hdx
```

（6）复制内存资料到硬盘。

将内存里的数据复制到 root 目录下的 mem.bin 文件。

```
dd if=/dev/mem of=/root/mem.bin bs=1024
```

要从光盘复制 iso 镜像，可复制光盘数据到 root 文件夹下，并保存为 cd.iso 文件。

```
dd if=/dev/cdrom of=/root/cd.iso
```

（7）增加 swap 分区文件大小。

创建一个足够大的文件（此处为 256MB）。

```
dd if=/dev/zero of=/swapfile bs=1024 count=262144
```

把这个文件变成 swap 文件。

```
mkswap /swapfile
```

启用这个 swap 文件。

```
swapon /swapfile
```

在每次开机的时候自动加载 swap 文件，需要在/etc/fstab 文件中增加一行。

```
/swapfile swap swap defaults 0 0
```

（8）销毁磁盘数据。

利用随机的数据填充硬盘。

```
dd if=/dev/urandom of=/dev/hda1
```

在某些必要的场合可以用来销毁数据。执行此操作以后，/dev/hda1 将无法挂载，创建和复制操作无法执行。

（9）修复硬盘。

当硬盘较长时间（比如一两年）放置不使用后，磁盘上会产生 magnetic flux point。当磁头读到这些区域时会遇到困难，并可能导致 I/O 错误。当这种情况影响到硬盘的第一个扇区时，可能导致硬盘报废。下面的命令有可能使这些数据起死回生。

```
dd if=/dev/sda of=/dev/sda
```

3.4 fdisk：磁盘分区

作用：fdisk 命令用来进行磁盘分区。

用法：fdisk [选项] device

主要选项如下。

- -l：查看指定设备的分区表状况。
- -b SSZ：将指定的分区大小输出到标准输出上，单位为区块。
- -u：搭配“-l”参数列表，会用分区数目取代柱面数目来表示每个分区的起始地址。
- device：进行这些操作的设备名称。

说明

fdisk 是各种 Linux 发行版本中最常用的分区工具，是被定义为 Expert 级别的分区工具。它还包括一个二级选单，首先输入命令，然后出现问答式界面，用户通过在这个界面中输入命令参数来操作 fdisk，如图 3-1 所示。

图 3-1　fdisk 帮助选单

选单详细介绍如下。

- a：设定硬盘启动区。
- b：编辑一个 BSD 类型分区。
- c：编辑一个 DOS 兼容分区。
- d：删除一个分区。
- l：查看指定设备的分区表状况。
- m：显示 fdisk 命令各个参数的说明。
- o：创建一个 DOS 分区。
- n：设定新的硬盘分区。
- p：打印分区信息。
- s：创建一个空 Sun 分区表。
- t：改变硬盘分区类型。
- q：结束分区，不保存操作内容。

– v：校验硬盘分区表。
– w：结束分区，保存操作内容。
– x：进入高级操作模式。

应用实例如下。

（1）查看所有硬盘及其分区情况。

```
# fdisk -l
Disk /dev/sda: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
   Device Boot     Start         End      Blocks   Id  System
/dev/sda1   *           1          13      104391   83  Linux
/dev/sda2              14        1044     8281507+  8e  Linux LVM
```

（2）使用 fdisk 为系统添加新硬盘。

以添加 IDE 接口硬盘为例，操作步骤如下。

① 物理安装。

注意

在新添加硬盘的时候，不要随便动以前的硬盘接口，否则系统就无法启动了。将硬盘挂在计算机上以后，启动计算机，能够进入 BIOS 中去查看主板有没有认出新加载的硬盘。

② 启动计算机到 Linux 下面，使用命令“fdisk －l”能够看到刚刚安装上去的硬盘，下一步对硬盘进行分区和格式化。

```
fdisk /dev/hhdb
```

③ 进入 fdisk 模式。

```
Command (m for help):p //查看新硬盘的分区
Command (m for help):n //创建新分区
Command action
e extended //输入 e 为创建扩展分区
p primary partition （1-4）//输入 p 为创建主分区，这里选择 p
Partion number（1-4）：1    //第一个扩展分区，按需求最多能够分 4 个主分区
First Cylinder(1-1014,default 1): 1 //第一个主分区起始的磁盘块数
Last cylindet or +siza or +sizeM or +sizeK: +1871MB //可以是以 MB 为单位的数字或磁盘块数，输入+1024MB 表示分区大小为 1GB
Command (m for help): w
```

```
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

操作过程如图 3-2 所示。

图 3-2　使用硬盘分区程序 fdisk

④ 建立好硬盘分区后，执行命令，对硬盘进行格式化操作。

```
# mke2fs /dev/hdb1
```

⑤ 挂载到目录。硬盘格式化之后，可利用 mount 命令将该硬盘分区挂载到一个目录上。先创建该目录，如要挂载的目录名为/disk2，依次执行“mkdir disk2”、“mount /dev hdb1/disk2”命令，就可以使用该新硬盘了。

⑥ 设置启动时自动挂载。按照上面的操作，每次计算机启动都必须执行 mount 命令进行挂载才可使用，这样做很麻烦，可以通过修改/etc/fstab 配置文件使计算机在启动时自动挂载该硬盘分区。修改完配置文件后重新启动计算机，系统就会自动挂载该分区。至此，就可以方便地使用添加的硬盘了。

3.5 mount：挂载文件系统

作用：挂载文件系统。

用法：mount[-lhV]

mount-a[选项]

mount[选项] [--source]<源>|[--target]<目录>

mount[选项]<源><目录>

mount<操作><挂载点>[<目标>]

主要选项如下。

- -a,--all：挂载 fstab 中的所有文件系统。
- -c,--no-canonicalize：不对路径规范化。
- -f,--fake：空运行；跳过 mount(2)系统调用。
- -F,--fork：对每个设备禁用 fork（和-a 选项一起使用）。
- -T,--fstab<路径>：/etc/fstab 的替代文件。
- -i,--internal-only：不调用 mount.<类型>助手程序。
- -l,--show-labels：列出所有带有指定标签的挂载。
- -n,--no-mtab：不写/etc/mtab。
- -o,--options<列表>：挂载选项列表，以英文逗号分隔。
- -O,--test-opts<列表>：限制文件系统集合（和-a 选项一起使用）。
- -r,--read-only：以只读方式挂载文件系统（同-o ro）。
- -t,--types<列表>：限制文件系统类型集合。
- --source<源>：指明源（路径、标签、UUID）。
- --target<目标>：指明挂载点。
- -v,--verbose：打印当前进行的操作。
- -V,--version：显示版本信息并退出。
- -w,--rw,--read-write：以读/写方式挂载文件系统（默认）。
- -h,--help：显示此帮助并退出。
- -V,--version：输出版本信息并退出。

源如下。

- -L,--label<标签>：同 LABEL=<label>。
- -U,--uuid<uuid>：同 UUID=<uuid>。
- LABEL=<标签>：按文件系统标签指定设备。
- UUID=<uuid>：按文件系统 UUID 指定设备。
- PARTLABEL=<标签>：按分区标签指定设备。
- PARTUUID=<uuid>：按分区 UUID 指定设备。
- <设备>：按路径指定设备。
- <目录>：绑定挂载的挂载点（参阅--bind/rbind）。
- <文件>：用于设置回环设备的常规文件。

操作如下。

- -B,--bind：挂载其他位置的子树（同-obind）。
- -M,--move：将子树移动到其他位置。
- -R,--rbind：挂载其他位置的子树及其包含的所有挂载。
- --make-shared：将子树标记为共享。
- --make-slave：将子树标记为从属。
- --make-private：将子树标记为私有。

- --make-unbindable：将子树标记为不可绑定。
- --make-rshared：递归地将整个子树标记为共享。
- --make-rslave：递归地将整个子树标记为从属。
- --make-rprivate：递归地将整个子树标记为私有。
- --make-runbindable：递归地将整个子树标记为不可绑定。

应用实例如下。

（1）USB 磁盘挂载。

mount 命令最重要的功能是挂载磁盘，目前使用的磁盘包括 IDE/SCSI 硬盘、IDE/SCSI 光盘、USB 磁盘。以 USB 磁盘为例，其挂载方法如下：

```
#fdisk -l        #查看 USB 磁盘设备号，一般是/dev/sda1#
#mkdir /mnt/usb  #建立一个挂载目录#
mount –t vfat /dev/sda1 /mnt/usb
```

（2）使用光盘驱动器。

一般情况下，只有超级用户可以使用 mount 命令，下面介绍如何让普通用户可以使用光盘驱动器。修改/etc/fstab 文件，加入如下一行：

```
#/dev/cdrom   /mnt/cdrom    udf,iso9660 noauto,owner,kudzu,ro，cao1 0 0
```

这样用户 cao1 也可以使用光盘驱动器了。

（3）查看已经加载的 VFAT 文件系统。

```
# mount -t vfat
```

（4）挂载/etc/fstab 配置文件中使用设备。

```
# mount -av
```

（5）网络文件系统挂载。

Linux 支持 NFS 网络文件系统，首先设置 NFS 网络文件系统，然后利用 mount 命令挂载 NFS 服务器的/home/cao 目录。

```
# mount -t nfs 192.168.1.4:/home/cao /home/nfs/cao
```

下面使用 df 命令查看挂载的目录，其中已经包括 NFS 服务器的共享目录，如图 3-3 所示。

图 3-3　使用 df 命令查看挂载的目录

3.6 umount：卸载文件系统

作用：umount 命令用来卸除文件系统。

用法：umount [-hV]

umount -a [选项]

umount [选项] <源> | <目录>

主要选项如下。

- -a,--all：卸载所有文件系统。
- -A,--all-targets：卸载当前名字空间内指定设备对应的所有挂载点。
- -c,--no-canonicalize：不对路径规范化。
- -d,--detach-loop：若挂载了回环设备，也释放该回环设备。
- --fake：空运行；跳过 umount(2)系统调用。
- -f,--force：强制卸载（遇到不响应的 NFS 系统时）。
- -i,--internal-only：不调用 umount.<类型>辅助程序。
- -n,--no-mtab：不写/etc/mtab。
- -l,--lazy：立即断开文件系统，所有清理以后执行。
- -O,--test-opts<列表>：限制文件系统集合（和-a 选项一起使用）。
- -R,--recursive：递归卸载目录及其子对象。
- -r,--read-only：若卸载失败，尝试以只读方式重新挂载。
- -t,--types<列表>：限制文件系统集合。
- -v,--verbose：打印当前进行的操作。
- -h,--help：显示此帮助并退出。
- -V,--version：输出版本信息并退出。

应用实例如下。

卸载一个 USB 盘。

```
#umount  /mnt/usb
```

说明

mount 加载设备的信息记录在/etc/mtab 文档中。使用 umount 命令卸载设备时，记录将被清除。

3.7 mkfs：建立各种文件系统

作用：mkfs 命令用来在 Linux 下建立各种文件系统。

用法：mkfs [选项] [-t <类型>] [文件系统选项] <设备> [<大小>]

主要选项如下。

- -t,--type=<类型>：文件系统类型；若不指定，将使用 Ext2。
- fs-options：实际文件系统构建程序的参数。
- <设备>：要使用设备的路径。
- <大小>：要使用设备上的块数。
- -V,--verbose：解释正在进行的操作；多次指定-V 将导致空运行（dry-run）。
- -V,--version：显示版本信息并退出；将-V 作为--version 选项时必须是唯一选项。
- -h,--help：显示此帮助并退出。

说明

mkfs 命令本身并不执行建立文件系统的工作，它是在 Linux 下各文件系统专用程序（mkfs.fstype）的前端程序。各文件系统专用程序可以在/sbin、/sbin/fs、/sbin/fs.d 等目录中找到。mkfs.dos 用来创建 DOS 文件系统；mkfs.reiserfs 用来创建 ReiserFS 文件系统；kfs.jfs 用来创建 JFS 文件系统；mkfs.vfat 用来创建 VFAT 文件系统。出于对 Linux 各个发行版本的兼容性，对于建立同一 Linux 文件系统的命令可能有不同的命令名称，实际功能是相同的，如 mkfs.ext2、mkfs.ext3、mke2fs 都可用来创建 Ext2/Ext3 文件系统。

应用实例如下。

（1）在设备/dev/hda8 上建立 Ext3 文件系统。

```
# mkfs -t ext3 /dev/hda8
mke2fs 1.35 （28-Feb-2004）
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
251392 inodes, 502023 blocks
25101 blocks (5.00%) reserved for the super user
First data block=0
16 block groups
32768 blocks per group, 32768 fragments per group
15712 inodes per group
Superblock backups stored on blocks:
32768, 98304, 163840, 229376, 294912
Writing inode tables: done
Creating journal (8192 blocks): done
```

```
Writing superblocks and filesystem accounting information: done
This filesystem will be automatically checked every 32 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
```

（2）在设备/dev/hda7 上创建 XFS 文件系统。

```
#mkfs -t xfs /dev/hda7
meta-data=/dev/hda7 isize=256 agcount=16, agsize=156382 blks
= sectsz=512
data = bsize=4096 blocks=2502112, imaxpct=25
= sunit=0 swidth=0 blks, unwritten=1
naming =version 2 bsize=4096
log =internal log bsize=4096 blocks=2560, version=1
= sectsz=512 sunit=0 blks
realtime =none extsz=65536 blocks=0, rtextents=0
```

3.8 mkfs.ext2：建立一个 Ext2/Ext3 文件系统

作用：mkfs.ext2 命令用来建立一个 Ext2/Ext3 文件系统。

用法：mkfs.ext2 [选项] device [blocks]

主要选项如下。

- -b block-size：指定区块大小，单位为字节。
- -f fragment-size：指定不连续区段的大小，单位为字节。
- -F：强制执行 mke2fs 命令。
- -c：检查是否有损坏的区块。
- -L volume-label：设置分区的卷标（最长 11 个字符）。
- -M last-mounted-directory：记录最后一次挂入的目录。
- -m reserved-blocks-percentage：指定给管理员保留区块的比例，默认值为 5%，这是 UNIX 默认值的一半。
- -N number-of-inodes：指定要建立的 inode 数目。
- -i bytes-per-inode：指定“字节/inode”的比例。
- -j：建立一个 Ext3 日志文件系统。
- -r fs-revision：指定要建立的 Ext2 文件系统版本。
- -l filename：从指定文件中读取文件系统损坏区块的信息。
- -R raid_opts：设置磁盘阵列参数。
- -q：执行时不显示任何信息。
- -v：执行时显示详细信息。
- device：设备名称。

– blocks：区块数。

应用实例如下。

（1）在第二个 SCSI 磁盘的第二个分区上创建一个 Ext2 文件系统。

```
# mkfs.ext2 /dev/sdb2
```

（2）检查第二个 IDE 磁盘第一个分区的坏块之后，创建一个文件系统，卷标设置为“cjh”，保留 2%的空间，创建 1000 个 inode。

```
#mke2fs -L cjh -c -q -m 2 -N 1000 /dev/hdb1
```

3.9 mkbootdisk：建立启动盘

作用：mkbootdisk 命令用来建立系统的启动盘。

用法：mkbootdisk [选项] [--device<devicefile>]

主要选项如下。

– --device<devicefile>：指定设备，默认是/dev/fd0。
– --mkinitrdargs<参数>：设置 mkinitrd 的参数。
– --noprompt：一般情况下 kbootdisk 会提示用户插入磁盘，使用此参数可省略此过程。
– --verbose：执行时显示详细的信息。
– --iso：mkbootdisk 制作一个 ISO 映象文件。
– --version-v：显示版本信息。
– --size <size>：指定启动文件大小，默认是 1.44MB。

应用实例如下。

动手快速制作一张启动盘，步骤如下。

① 查看内核版本。

```
#uname –r
2.6.20-8
```

② 自动完成启动盘制作。

```
#mkbootdisk --device /dev/fd0 --noprompt 2.6.20-8
```

如果没有任何错误提示，就表示操作成功。

3.10 fsck：检查文件系统

作用：fsck 命令用来检查文件系统并尝试修复错误。

用法：fsck [选项] [-t fstype] device

主要选项如下。

- -a：自动修复文件系统，不询问任何问题。
- -A：依照/etc/fstab 配置文件的内容，检查文件内所列的全部文件系统。
- -N：不执行指令，仅列出实际执行会进行的动作。
- -P：当搭配“-A”参数使用时，会同时检查所有的文件系统。
- -r：采用互动模式，在执行修复时询问问题，让用户得以确认并决定处理方式。
- -R：当搭配“-A”参数使用时，会略过“/”目录的文件系统不予检查。
- -s：依次执行检查作业，而非同时执行。
- -t fstype：指定要检查的文件系统类型。
- -T：执行 fsck 指令时，不显示标题信息。
- -V：显示指令执行过程。

说明

fsck 是 Linux 各文件系统检查程序（fsck.fstype）的前端程序。出于对 Linux 各个发行版本的兼容性，检查 Ext2/Ext3 文件系统的命令可能有不同的命令名称，实际功能是相同的，如 fsck.Ext2、fsck.Ext3、E2fsck 都用来检查 Ext2/Ext3 文件系统。

e2fsck 执行的传回值及代表意义如下。

- 0：没有任何错误发生。
- 1：文件系统发生错误，并且已经修正。
- 2：文件系统应当重启。
- 4：文件系统发生错误，但没有修正。
- 8：运行时发生错误。
- 16：使用的语法发生错误。
- 128：共享的函数库发生错误。

应用实例如下。

检查设备/dev/sda1。

```
# fsck /dev/sda1
```

3.11 blockdev：从命令行调用区块设备控制程序

作用：在命令行调用“ioxtls”函数，以实现对设备的控制。

用法：blockdev -V

blockdev –report [设备]

blockdev [-v|-q] 命令 设备

主要选项如下。

- -V：打印版本号并退出。
- -q：安静模式。
- -v：详细信息模式。
- --report：打印报告指定设备。

可用命令如下。

- --getsz：获得 512 字节扇区的大小。
- --setro：设置只读。
- --setrw：设置读/写。
- --getro：获得只读。
- --getdiscardzeroes：获取忽略零数据支持状态。
- --getss：获得逻辑块（扇区）大小。
- --getpbsz：获得物理块（扇区）大小。
- --getiomin：获得最小 I/O 大小。
- --getioopt：获得最优 I/O 大小。
- --getalignoff：获得对齐偏移字节数。
- --getmaxsect：获得每次请求的最大扇区数。
- --getbsz：获得块大小。
- --setbsz<bytes>：设置块设备大小。
- --getsize：获得 32bit 扇区数量（已废弃，请使用--getsz）。
- --getsize64：获得字节大小。
- --setra<sectors>：设置 readahead。
- --getra：获取 readahead。
- --setfra<sectors>：设置文件系统 readahead。
- --getfra：获取文件系统 readahead。
- --flushbufs：刷新缓存。
- --rereadpt：重新读取分区表。

应用实例如下。

（1）设置设备为只读。

```
# blockdev --setro /dev/hda4
```

（2）读取设备是否为只读。

```
#blockdev --getro /dev/hda4
```

3.12 hdparm：设置磁盘参数

作用：hdparm 命令用来检测、显示与设定硬盘的参数。

用法：hdparm[选项] [设备]

主要选项如下。

- -a：设定读取文件时，预先存入块区的分区数。
- -A：启动或关闭读取文件时的快取功能，可选 0 或 1。
- -c：设定 IDE 硬盘 32 位 I/O 模式。
- -C：检测 IDE 硬盘的电源管理模式。
- -d：设定磁盘的 DMA 模式，可选 0 或 1。
- -f：将内存缓冲区的数据写入硬盘，并清空缓冲区。
- -g：显示硬盘的磁轨、磁头、磁区等参数。
- -h：显示帮助信息。
- -i：显示硬盘的硬件信息，这些信息是在开机时由硬盘本身所提供的。
- -I：直接读取硬盘所提供的硬件规格信息。
- -k：重新设定硬盘参数时，保留-dmu 参数的设定，可选 0 或 1。
- -K：重新设定硬盘参数时，保留-APSWXZ 参数的设定，可选 0 或 1。
- -m<扇区数目>：设定硬盘多重分区存取的分区数。
- -n：是否忽略硬盘写入时所发生的错误，可选 0 或 1。
- -p<PIO 模式>：设定硬盘的 PIO 模式，可选 0、1、2、3、4、5。
- -P<扇区数目>：设定硬盘内部快速存取的分区数。
- -q：执行后续的参数时，不在屏幕上显示任何信息。
- -r：设定硬盘的读/写模式，可选 0 或 1。
- -S：设定硬盘进入省电模式前的等待时间。
- -t：检测硬盘的读取数据效率。
- -T：检测硬盘快速存取（即寻道时间）的读取效率。
- -u：在硬盘存取数据时，是否允许其他中断要求同时执行，可选 0 或 1。
- -v：显示硬盘的相关设定。
- -W：设定硬盘的写入快速存取，可选 0 或 1。
- -X<传输模式>：设定硬盘的传输模式。
- -y：使 IDE 硬盘进入省电模式。
- -Y：使 IDE 硬盘进入睡眠模式。
- -Z：关闭 Seagate 硬盘的自动省电功能。
- -z：重新读取硬盘分区表。

说明

hdparm 命令是进行硬盘性能调谐的一个最常用的命令，使用这个命令前，需要对硬盘的硬件知识有一定了解，另外需要做好数据备份。

应用实例如下。

使用 hdparm 命令来调谐磁盘性能，步骤如下。

① 查看默认设置。

```
#hdparm –v /dev/hda
```

② 测试并且记录默认设置的磁盘性能。

```
#hdparm –tT /dev/hda
```

③ 重新设定硬盘各项参数。

```
#hdparm -u1 -d1 -m16 -c3 -X66 /dev/hda
```

- -u1：Linux 在处理磁盘中断时，可以响应其他中断相关的任务。
- -d1：打开 DMA 模式。
- -m16：改变硬盘的多路扇区的读功能。-m16 可以使得硬盘在一次 I/O 中断中读入 16 个扇区的数据。
- -c3：把硬盘的 16 位模式转换为 32 位模式（32 bit mode w/sync）。
- -X66：启用 DMA66 数据传输模式。

④ 记录修改设置后的磁盘性能，对比数据。如果磁盘的性能有所改进，则保存设置。

```
#hdparm –t1 /dev/hda
```

3.13 mkswap：建立交换分区

作用：设置交换区（swap area）。

用法：mkswap [选项] [设备名称或文件] [交换区大小]

主要选项如下。

- -c：建立交换区前，先检查是否有损坏的区块。
- -f：在 SPARC 计算机上建立交换区时，要加上此参数。
- -v0：建立旧式交换区，此为预设值。
- -v1：建立新式交换区。
- [交换区大小]：指定交换区的大小，单位为 1024 字节。

说明

mkswap 可将磁盘分区或文件设为 Linux 的交换区。在一个文件或者设备上建立交换区，在建立完之后要使用 swapon 指令开始使用这个交换区。最后一个选择性参数指定了交换区的大小，但是这个参数是为了后向兼容设置的，没有使用的必要。一般都将整个文件或设备作为交换区。

应用实例如下。

指定一个设备作为交换区。

```
# mkswap -c -v1 -L swapnew /dev/hda3
```

3.14 swapon：使用交换空间

作用：swapon 命令用于激活 Linux 系统中的交换空间，Linux 系统的内存管理必须使用交换区来建立虚拟内存。

用法：swapon [选项] [<指定>]

主要选项如下。

- -a,--all：启用/etc/fstab 中的所有交换区。
- -d,--discard：在释放的页重新使用前忽略它们。
- -e,--ifexists：直接跳过不存在的设备。
- -f,--fixpgsz：需要时重新初始化交换空间。
- -p,--priority<prio>：指定交换设备的优先级为<prio>。
- -s,--summary：显示已使用交换设备的摘要。
- --show[=<列>]：定义所显示摘要表格中的列。
- --noheadings：不打印标题，与--show 一起使用。
- --raw：使用原生输出格式，与--show 一起使用。
- --bytes：在--show 选项输出中以字节显示交换区大小。
- -v,--verbose：详尽模式。
- -h,--help：显示此帮助并退出。
- -V,--version：输出版本信息并退出。

<spec>参数如下。

- -L<标签>：同 LABEL=<标签>。
- -U<uuid>：同 UUID=<uuid>。
- LABEL=<标签>：按交换区标签指定设备。
- UUID=<uuid>：按交换区 UUID 指定设备。
- PARTLABEL=<标签>：按分区标签指定设备。
- PARTUUID=<uuid>：按分区 UUID 指定设备。
- <设备>：要使用设备的名称。
- <文件>：要使用文件的名称。

可用列（用于--show）如下。

- NAME：设备文件或分区路径。
- TYPE：设备的类型。
- SIZE：交换区大小。

– USED：已使用字节数。
– PRIO：交换优先级。

应用实例如下。

（1）创建大小为 512MB 的交换文件。

```
# dd if=/dev/zero of=/swapfile1 bs=1024 count=524288  [root@linux ~]
```

（2）使用 mkswap 命令设置交换文件。

```
# mkswap /swapfile1
# 启用交换分区
# swapon /swapfile1
```

3.15 swapoff：关闭系统交换分区

作用：swapoff 命令用于关闭交换分区，swapon 命令用于打开交换分区。

用法：swapoff [选项] [<指定>]

主要选项如下。

– -a,--all：禁用/proc/swaps 中的所有交换区。
– -v,--verbose：显示详细输出信息。
– -h,--help：显示此帮助并退出。
– -V,--version：输出版本信息并退出。

<指定>参数主要有如下几个。

– -L<标签>：要使用设备的标签。
– -U<uuid>：要使用设备的 UUID。
– LABEL=<标签>：要使用设备的标签。
– UUID=<uuid>：要使用设备的 UUID。
– <设备>：要使用设备的名称。
– <文件>：要使用文件的名称。

应用实例如下。

关闭交换分区。

```
# swapoff /dev/sda2
```

3.16 sync：写入磁盘

作用：将内存缓冲区内的数据写入磁盘。

用法：sync

说明

在 Linux 系统中，当数据需要存入磁盘时，通常会先放到缓冲区内，等到适当的时刻再写入磁盘，如此可提高系统的执行效率。本命令无参数。

3.17 e2label：设置卷标

作用：设定或显示 Ext2 或 Ext3 分区的卷标。

用法：e2label device [new-label]

主要选项如下。

- device：分区设备名称。
- new-label：新设定的卷标名称，不能超过 16 个字符。

应用实例如下。

为一个新创建的分区建立卷标名称 goodcjh。

```
#e2label /dev/hda9 goodcjh
```

3.18 badblocks：检查磁盘

作用：检查磁盘中损坏的区块。

用法：badblocks [选项] device [last_block [start_count]

主要选项如下。

- -b block_size：指定磁盘的区块大小，单位为字节。
- -i input_file：读出上次检查结果中的坏块列表。
- -o output_file：将检查的结果写入指定的输出文件。
- -s：在检查时显示进度。
- -v：在执行时显示详细的信息。
- -w：在检查时执行写入测试。
- -c blocks_at_once：每个区块检查的次数，默认值为 16 次。
- -p num_passes：通过的数量，默认值为 0。
- start_count：指定要从哪个区块开始检查。
- last_block：指定结束检查的区块。
- device：指定要检查的磁盘装置。

应用实例如下。

以 4096 字节的一个区块（Block）为例，每个区块检查 16 次，将结果输入 badblocks-list（badblocks-list 是一个文本文件）。

```
#badblocks -b 4096 /dev/hda1 -o badblocks-list
```

如果硬盘正常的话，badblocks-list 中应该是没有任何内容的。

3.19 quota：显示磁盘已使用的空间与限制

作用：显示磁盘已使用的空间与限制。

用法：quota [选项] [用户名称...]

quota [选项] [群组名称...]

主要选项如下。

- -g：列出群组的磁盘空间限制。
- -q：简明列表，只列出超过限制的部分。
- -u：列出用户的磁盘空间限制。
- -v：显示该用户或群组所有挂入系统的存储设备的空间限制。
- -V：显示版本信息。

> **说明**
>
> 执行 quota 指令，可查询磁盘空间的限制，并可知已使用多少空间。

应用实例如下。

显示 cjh 组磁盘已使用的空间。

```
# quota cjh
```

3.20 quotacheck：检查磁盘的使用空间与限制

作用：检查磁盘的使用空间与限制。

用法：quotacheck [-选项] [文件系统...]

主要选项如下。

- -a：扫描在/etc/fstab 文件中加入 quota 设置的分区。
- -d：详细显示指令执行过程，便于排错或了解程序执行的情形。
- -g：扫描磁盘空间时，计算每个群组识别码所占用的目录和文件数目。
- -R：排除根目录所在的分区。

– -u：扫描磁盘空间时，计算每个用户识别码所占用的目录和文件数目。
– -v：显示指令执行过程。

说明

执行 quotacheck 命令会扫描挂入系统的分区，并在各分区的文件系统根目录下产生 quota.user 和 quota.group 文件，分别用于设置用户和群组的磁盘空间限制。

应用实例如下。

扫描在/etc/fstab 文件中加入 quota 设置的分区。

```
# quotacheck -a
```

3.21 quotaoff：关闭磁盘空间限制

作用：关闭磁盘空间限制。

用法：quotaoff [选项] [文件系统...]

主要选项如下。

– -a：关闭在/etc/fstab 文件中加入 quota 设置的分区的空间限制。
– -g：关闭群组的磁盘空间限制。
– -u：关闭用户的磁盘空间限制。
– -v：显示指令执行过程。

说明

执行 quotaoff 指令可以关闭用户和群组的磁盘空间限制。

应用实例如下。

关闭在/etc/fstab 文件中加入 quota 设置的分区的空间限制。

```
# quotaoff -a
```

3.22 quotaon：开启磁盘空间限制

作用：开启磁盘空间限制。

用法：quotaon [选项] [文件系统...]

主要选项如下。

- -a：开启在/ect/fstab 文件中加入 quota 设置的分区的空间限制。
- -g：开启群组的磁盘空间限制。
- -u：开启用户的磁盘空间限制。
- -v：显示指令执行过程。

> **说明**
>
> 执行 quotaon 指令可开启用户和群组的磁盘空间限制，各分区的文件系统根目录必须有 quota.user 和 quota.group 配置文件。

应用实例如下。

开启在/etc/fstab 文件中加入 quota 设置的分区的空间限制。

```
# quotaon -a
```

3.23 quotastats：显示磁盘空间限制

作用：显示磁盘空间限制。

用法：quotastats

> **说明**
>
> 本命令无参数。

应用实例如下。

显示磁盘空间限制。

```
# quotastats
Kernel quota version: 6.5.1
Number of dquot lookups: 0
Number of dquot drops: 0
Number of dquot reads: 0
Number of dquot writes: 0
Number of quotafile syncs: 22
Number of dquot cache hits: 0
Number of allocated dquots: 0
Number of free dquots: 0
Number of in use dquot entries (user/group): 0
```

3.24 repquota：检查磁盘空间限制的状态

作用：检查磁盘空间限制的状态。

用法：repquota [选项] [文件系统...]

主要选项如下。

- -a：列出在/etc/fstab 文件中加入 quota 设置的分区的使用状况，包括用户和群组。
- -g：列出所有群组的磁盘空间限制。
- -u：列出所有用户的磁盘空间限制。
- -v：显示该用户或群组的所有空间限制。

说明

执行 repquota 指令可以报告磁盘空间限制的状态，清楚得知每位用户或每个群组已使用多少空间。

应用实例如下。

显示 cjh 用户组的磁盘使用情况。

```
#  repquota -g cjh
```

3.25 mdadm：RAID 设置工具

作用：mdadm 是 Linux 下用于管理软件磁盘阵列的工具。

用法：mdadm [模式] [设备] [选项]

模式有如下 7 种。

- Create：使用空闲的设备创建一个新的阵列，每个设备具有元数据块。
- Assemble：将原来属于一个阵列的每个块设备组装为阵列。
- Build：创建或组装不需要元数据的阵列，每个设备没有元数据块。
- Manage：管理已经存储在阵列中的设备，比如增加热备份磁盘或者设置某个磁盘失效，然后从阵列中删除这个磁盘。
- Misc：报告或者修改阵列中相关设备的信息，比如查询阵列或者设备的状态信息。
- Grow：改变阵列中每个设备被使用的容量或阵列中的设备数目。
- Monitor：监控一个或多个阵列，上报指定的事件。

主要选项如下。

- -A, --assemble：加入一个以前定义的磁盘阵列。
- -B, --build：建立没有超级块（Super Block）的磁盘阵列。

- -C, --create：创建一个新的磁盘阵列。
- -Q, --query：查看一个设备。
- -D, --detail：显示一个或多个磁盘阵列的详细信息。
- -E, --examine：显示一个或多个磁盘阵列的超级块信息。
- -F, --follow, --monitor：选择 Monitor 模式。
- -G, --grow：改变在用磁盘阵列的大小或形态。
- -h, --help：帮助信息。用在以上选项后，则显示该选项信息。
- -c, --config=：指定配置文件，默认为/etc/mdadm/mdadm.conf。
- -s, --scan：扫描配置文件或/proc/mdstat，以搜寻丢失的信息。

mdadm --create 主要参数如下。

- --auto=yes：自动连接的磁盘阵列设备，一般是/dev/md0，以此类推是/dev/md1……
- --raid-devices=N：使用几个磁盘（Partition）作为磁盘阵列的设备。
- --spare-devices=N：使用几个磁盘作为磁盘阵列的备用设备。
- --level=[015]：设置磁盘阵列的等级，常用 0,1,5。

mdadm --manage 主要参数如下。

- --add：将后面的设备加入到这个 MD 中。
- --remove：将后面的设备由这个 MD 中移除。
- --fail：将后面的设备设定为出错的状态。

应用实例如下。

1）创建新的磁盘

mdadm 使用--create（或者其缩写-C）参数来创建新的阵列，并且将一些重要阵列的标识信息作为元数据，可以写在每一个底层设备的指定区间。--level（或者其缩写-l）表示阵列的 RAID 级别。--chunk（或者其缩写-c）表示每个条带单元的大小，以 KB 为单位，默认为 64KB，条带单元的大小配置对不同负载下的阵列读/写性能有很大影响。--raid-devices（或者其缩写-n）表示阵列中活跃的设备个数。而--spare-devices（或者其缩写-x）表示阵列中热备磁盘的个数，一旦阵列中的某个磁盘失效，MD 内核驱动程序自动将热备磁盘加入到阵列，然后将丢失磁盘上的数据重构到热备磁盘上。

创建一个 RAID 0 设备。

```
mdadm --create  /dev/md0 --level=0 --chunk=32 --raid-devices=3 /dev/sd[i-k]1
```

创建一个 RAID 1 设备。

```
mdadm -C /dev/md0 -l1 -c128 -n2 -x1 /dev/sd[i-k]1
```

创建一个 RAID 5 设备。

```
mdadm -C /dev/md0 -l5 -n5 /dev/sd[c-g] -x1 /dev/sdb
```

创建一个 RAID 6 设备。

```
mdadm -C /dev/md0 -l6 -n5 /dev/sd[c-g] -x2 /dev/sdb /dev/sdh
```

创建一个 RAID 10 设备。

```
mdadm -C /dev/md0 -l10 -n6 /dev/sd[b-g] -x1 /dev/sdh
```

创建一个 RAID 1+0 设备。

```
mdadm -C /dev/md0 -l1 -n2 /dev/sdb /dev/sdc
mdadm -C /dev/md1 -l1 -n2 /dev/sdd /dev/sde
mdadm -C /dev/md2 -l1 -n2 /dev/sdf /dev/sdg
mdadm -C /dev/md3 -l0 -n3 /dev/md0 /dev/md1 /dev/md2
```

当 RAID 1/4/5/6/10 等创建成功后，需要计算每个条带的校验和信息，并写入到相应磁盘上，所以 RAID 阵列有一个冗余组数据同步的初始化过程（resync）。但是 MD 设备只要创建成功后即可对外被上层应用读/写使用，但是上层数据读/写会降低数据同步的性能。初始化的时间长短和磁盘阵列自身性能及读/写的应用负载有关，可以使用 cat /proc/mdstat 信息查询 RAID 阵列当前重构的速度和预期的完成时间。

```
# cat /proc/mdstat
Personalities : [raid10]
md0 : active raid10 sdh[6](S) sdg[5] sdf[4] sde[3] sdd[2] sdc[1] sdb[0]
     3145536 blocks 64K chunks 2 near-copies [6/6] [UUUUUU]
     [===>...........]  resync = 15.3% (483072/3145536) finish=0.3min
speed=120768K/sec
unused devices: <none>
```

2）使用磁盘

MD 设备可以像普通块设备那样直接读/写，也可以当作文件系统使用。

```
#mkfs.ext3 /dev/md0
#mkdir -p /mnt/md-test
#mount /dev/md0 /mnt/md-test
```

3）停止正在运行的阵列

如果阵列没有文件系统或其他存储应用及高级设备使用，可以使用--stop（或其缩写-S）停止阵列；如果命令返回设备或者资源忙类型的错误，说明/dev/md0 正在被上层应用使用，暂时不能停止，必须首先停止上层的应用，这样也能保证阵列上数据的一致性。

```
# ./mdadm --stop /dev/md0
mdadm: fail to stop array /dev/md0: Device or resource busy
# umount /dev/md0
#./mdadm --stop /dev/md0
mdadm: stopped /dev/md0
```

3.26 tune2fs：文件系统调整

作用：tune2fs 是 Linux 下重要的文件系统调整工具。

用法：tune2fs [选项]设备

主要选项如下。

- c max_mount_counts：用来调整最大挂接次数，如果到达这个次数，系统将执行文件系统检查。
- -C mount_count：用来设置文件系统已经挂接的次数。
- -i check_interval[d|m|w]：用来设置文件系统的最长使用时间，如果到达这个时间，系统将会执行文件系统检查。把这个值设置为 0 的时候，则参数不生效。
- -e error_behavior：下面列出 3 种发现错误后的行为。在默认情况下，如果系统检测到文件系统有错误，会设置文件系统在下次启动的时候执行 fsck 检测。
  - continue：继续执行检测。
  - remount-ro：重新以只读方式挂接。
  - panic：产生一次系统崩溃（Panic）。
- -l：显示文件系统的详细参数。
- -j：转换为 Ext3 文件系统，包括日志。
- -m：设置保留的空间百分比。
- -o：设置默认加载参数。

说明

tune2fs 是一个用来调整 Ext2 和 Ext3 文件系统的命令。在调整文件系统设置的时候，这个文件系统不能被挂接在系统上。除此以外，有些文件系统的设置会立即生效，而有些调整需要重新启动系统才能生效。

应用实例如下。

（1）调整每两次完整地检查根文件系统的时间间隔为 5 周。

```
# tune2fs -i 5w /dev/hda1
```

（2）系统保留空间也可以用 tune2fs 调整，如调整根文件系统保留空间为 40 000 个磁盘块。

```
# tune2fs -r 40000 /dev/hda1
```

（3）通常，如果使用 Ext3 文件系统，使用-c 0 关掉 mount 次数达到后的文件系统检查。

```
#tune2fs -m 10 /dev/sda1
#tune2fs -o acl,user_xattr /dev/sda1
#tune2fs -i 0 -c0 /dev/sda1
```

3.27 mkisofs：建立 ISO 9660 映象文件

作用：mkisofs 可将指定的目录与文件做成 ISO 9660 格式的映象文件，以供刻录光盘。

用法：mkisofs[选项] [目录或文件]

主要选项如下。

- -a 或--all：mkisofs 通常不处理备份文件。使用此参数可以把备份文件加入到映象文件中。
- -abstract<摘要文件>：指定摘要文件的文件名。
- -b<开机映象文件>或-eltorito-boot<开机映象文件>：指定在制作可开机光盘时所需的开机映象文件。
- -biblio<ISBN 文件>：指定 ISBN 文件的文件名，ISBN 文件位于光盘根目录下，记录光盘的 ISBN。
- -c<开机文件名称>：制作可开机光盘时，mkisofs 会将开机映象文件中的-eltorito-catalog<开机文件名称>全部内容做成一个文件。
- -C<盘区编号,盘区编号>：将许多节区合成一个映象文件时，必须使用此参数。
- -copyright<版权信息文件>：指定版权信息文件的文件名。
- -d 或-omit-period：省略文件后的句号。
- -f 或-follow-links：忽略符号链接。
- -hide<目录或文件名>：使指定的目录或文件在 ISO 9660 或 Rock Ridge Extensions 的系统中隐藏。
- -hide-joliet<目录或文件名>：使指定的目录或文件在 Joliet 系统中隐藏。
- -J 或-joliet：使用 Joliet 格式的目录与文件名称。
- -l 或-full-iso9660-filenames：使用 ISO 9660 32 字符长度的文件名。
- -L 或-allow-leading-dots：允许文件名的第一个字符为句号。
- -log-file<记录文件>：在执行过程中若有错误信息，预设会显示在屏幕上。
- -m<目录或文件名>或-exclude<目录或文件名>：指定的目录或文件名将不会放入映象文件中。
- -M<映象文件>或-prev-session<映象文件>：与指定的映象文件合并。
- -N 或-omit-version-number：省略 ISO 9660 文件中的版本信息。
- -o<映象文件>或-output<映象文件>：指定映象文件的名称。
- -p<数据处理人>或-preparer<数据处理人>：记录光盘的数据处理人。
- -print-size：显示预估的文件系统大小。
- -quiet：执行时不显示任何信息。
- -r 或-rational-rock：使用 Rock Ridge Extensions，并开放全部文件的读取权限。
- -R 或-rock：使用 Rock Ridge Extensions。
- -sysid<系统 ID>：指定光盘的系统 ID。
- -T 或-translation-table：建立文件名的转换表，适用于不支持 Rock Ridge Extensions 的系统中。

- -v 或-verbose：执行时显示详细的信息。
- -V<光盘 ID>或-volid<光盘 ID>：指定光盘的卷册集 ID。
- -volset-size<光盘总数>：指定卷册集所包含的光盘张数。
- -volset-seqno<卷册序号>：指定光盘片在卷册集中的编号。
- -x<目录>：指定的目录将不会放入映象文件中。
- -z：建立通透性压缩文件的 SUSP 记录，此记录目前只在 Alpha 机器上的 Linux 中有效。

应用实例如下。

从/home/cjh 目录建立一个映象文件，文件名为 goodcjh：

```
# #  mkisofs -o goodcjh /home/cjh/
INFO:   UTF-8 character encoding detected by locale settings.
        Assuming UTF-8 encoded filenames on source filesystem,
        use -input-charset to override.
Total translation table size: 122333
Total rockridge attributes bytes: 111122
Total directory bytes: 8192111
Path table size(bytes): 701111
Max brk space used 2100011
366 extents written (111 MB)
```

3.28 cfdisk：磁盘分区

作用：cfdisk 是用来进行磁盘分区的程序，它类似于 DOS 下的 fdisk，具有交互式操作界面，而非传统 fdisk 的问答式界面，用户可以轻易地利用方向键来操控分区操作。

用法：cfdisk [选项]设备

主要选项如下。

- -c<柱面数目>：忽略 BIOS 的数值，直接指定磁盘的柱面数目。
- -P：显示分区表的内容，附加参数“r”会显示整个分区表的详细资料，附加参数“s”会依照磁区的顺序显示相关信息，附加参数“t”则会以磁头、磁区、柱面的方式来显示资料。
- -s<磁区数目>：忽略 BIOS 的数值，直接指定磁盘的磁区数目。
- -v：显示版本信息。
- -a：使用箭头代替高亮。
- -z：以空分区表启动，而非从磁盘读取分区表。
- -c C：覆盖内核检测到的柱面数目。
- -h H：覆盖内核检测到的磁头数目。
- -s S：覆盖内核检测到的扇区数。

说明

cfdisk 也是一款不错的分区工具，在有些发行版中，此工具已经从 util-linux 包中剔除。cfdisk 具有简单易用的特点，和 DOS 中的 fdisk 相似。cfdisk 命令使用 ncurses 库函数提供了比 fdisk 更图形化的界面。目前在 Slackware、Debian 等主流发行版中存在 cfdisk，而 Redhat 系列把这个小工具剔除了（用户可以自己下载安装使用）。

应用实例如下。

（1）使用 cfdisk 进行分区。

```
#cfdisk /dev/had
```

接下来进入操作界面，如图 3-4 所示。

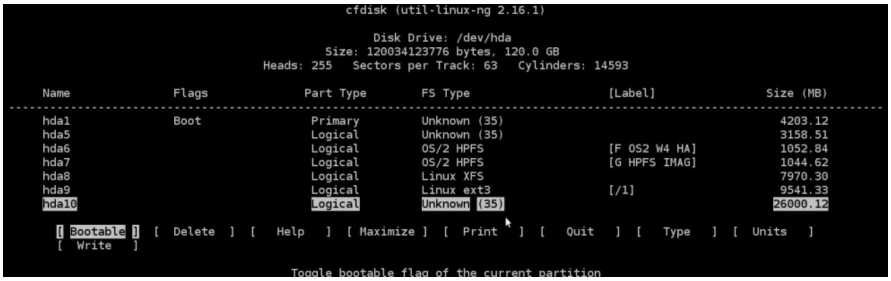


图 3-4 使用 cfdisk 分区

进入 cfdisk 操作界面后是基于鼠标操作的，操作命令如下。

- 上下箭头：切换分区。
- B：将分区设为可引导。
- D：删除当前分区。
- H 或?：显示在线说明。
- Q：改变柱面、磁头及扇区数。
- M：将分区的容量尽可能地最大化，但可能不兼容于 DOS 或 OS/2 操作系统。
- N：若还有未分配的分区，则建立新的分区。
- P：输出当前分区的状态。
- Q：不保存分区更改后的状态，而是结束程序。
- I：改变分区文件系统的类型。
- U：改变显示分区容量时的单位，可显示的单位有兆字节、柱面数及扇区数。
- W：写入分区更改后的状态。由于此操作影响较大，所以会紧接着要求确定，可输入“yes”确认或输入“no”取消。

（2）在程序里不用反白代表选取，而以箭头表示。

```
#cfdisk -a /dev/had
```

3.29 sfdisk：硬盘分区工具程序

作用：许多 Linux 发行版包含 sfdisk 命令。sfdisk 命令供程序员使用，可以进行脚本编程。只有确实知道正在做什么，才应该使用它。普通用户慎用。

用法：sfdisk [选项]设备

主要选项如下。

- -d：显示硬盘分区的设置。
- -g：显示硬盘的 CHS 参数。
- -l：显示每个硬盘的分区表信息。
- -s<分区>：显示分区的大小，单位为区块。
- -T 或--list-types：显示所有 sfdisk 能辨识的文件系统 ID。
- -v 或--version：显示版本信息。
- -V：检查硬盘分区是否正常。
- -x 或--show-extend：显示扩展分区中的逻辑分区。

应用实例如下。

（1）显示所有硬盘大小。

```
#  sfdisk -s
/dev/sda:   8388608
total: 8388608 blocks
```

（2）显示/dev/sda 设备大小。

```
# sfdisk -s /dev/sda
8388608
```

（3）显示/dev/sda 设备详细情况。

```
# sfdisk -l /dev/sda
Disk /dev/sda: 1044 cylinders, 255 heads, 63 sectors/track
Units = cylinders of 8225280 bytes, blocks of 1024 bytes, counting from 0
   Device Boot Start     End   #cyls    #blocks   Id  System
/dev/sda1   *      0+     12      13-    104391   83  Linux
/dev/sda2         13    1043    1031    8281507+  8e  Linux LVM
/dev/sda3          0       -       0          0    0  Empty
/dev/sda4          0       -       0          0    0  Empty
```

（4）对/dev/sda 设备进行检查。

```
# sfdisk -V /dev/sda
/dev/sda: OK
```

3.30 parted：磁盘分区工具

作用：管理、维护磁盘的分区。

用法：parted [选项]... [设备 [命令 [参数]...]...]

parted 命令将带有“参数”的命令应用于“设备”。如果没有给出“命令”，则以交互模式运行。

主要选项如下。

- -h,-help：显示此求助信息。
- -i,-interactive：在必要时，提示用户。
- -s,-script：从不提示用户。
- -v,-version：显示版本信息。

命令如下。

- 检查 MINOR：对文件系统进行一个简单的检查。
- cp [FROM-DEVICE] FROM-MINOR TO-MINOR：将文件系统复制到另一个分区。
- help [COMMAND]：打印通用求助信息，或关于 COMMAND 的信息。
- mklabel 标签类型：创建新的磁盘标签(分区表)。
- mkfs MINOR 文件系统类型：在 MINOR 创建类型为“文件系统类型”的文件系统。
- mkpart 分区类型 [文件系统类型] 起始点 终止点：创建一个分区。
- mkpartfs 分区类型 文件系统类型 起始点 终止点：创建一个带有文件系统的分区。
- move MINOR 起始点 终止点：移动编号为 MINOR 的分区。
- name MINOR 名称：将编号为 MINOR 的分区命名为“名称”。
- print [MINOR]：打印分区表，或者分区。
- quit：退出程序。
- rescue 起始点 终止点：挽救临近“起始点”、“终止点”的遗失的分区。
- resize MINOR 起始点 终止点：改变位于编号为 MINOR 的分区中的文件系统大小。
- rm MINOR：删除编号为 MINOR 的分区。
- select 设备：选择要编辑的设备。
- set MINOR 标志 状态：改变编号为 MINOR 的分区的标志。

说明

要启动 parted，使用命令 parted /dev/hdb，这里的/dev/hdb 是想配置的设备路径名称。

当看到一个“parted”提示时，输入 help 命令可以查看可用的命令，如表 3-1 所示。

表 3-1　parted 命令

命　　令	描　　述
check minor-num	执行文件系统的简单检查
cp from to	把文件系统从一个分区复制到另一个分区，from 和 to 是分区的次要号码
Help	显示可用的命令列表
mklabel label	为分区表创建磁盘标签
mkfs minor-num file-system-type	创建类型为 file-system-type 的文件系统
mkpart part-type fs-type start-mb end-mb	不创建新文件系统，而是制作分区
mkpartfs part-type fs-type start-mb end-mb	制作分区并创建指定的文件系统
move minor-num start-mb end-mb	移动分区
name minor-num name	仅为 MAC 和 PC 98 磁盘标签的分区命名
Print	显示分区表
rescue start-mb end-mb	拯救一个丢失的分区，从“start-mb”到“end-mb”
resize minor-num start-mb end-mb	重新划分分区大小，从“start-mb”到“end-mb”
rm minor-num	删除分区
select device	选择另一个设备来配置
set minor-num flag state	在分区上设置标志：state 要么是 on，要么是 off
Quit	退出 parted

说明

Linux 下磁盘分区管理的高级工具 parted 几乎可以完成以上所有命令的功能。许多 Linux 发行版本中都包括 parted。也可以在它的主页 http://www.gnu.org/software/parted/下载，最新版本为 1.6.22。

应用实例如下。

使用 parted 命令增大一个分区空间，条件是：该分区后边必须有未使用的空间，而且这个分区没有挂载。

首先使用“fdisk -l”命令查看分区现有情况。

```
# fdisk -l
…
/dev/hda9         269      750   3871633+  83  Linux
```

可以看到分区 9 从第一个 IDE 硬盘柱面 269 开始至 750 结束，大小是 3 871 633 KB，文件系统是 Ext3。如果要把分区 9 的大小从 3 871 633 KB 增加到 8 248 000 KB，只要把分区 9 结束位置从 750 调整至柱面 1300 即可。调整之前使用 umount 命令来卸除文件系统，然后执行如下命令：

```
parted /dev/hda resize 9269 1300
```

调整后重新启动计算机，分区的改变即可生效。

其他分区工具简介如下。

- qtparted 和 gtparted。qtparted 工具基于 parted，提供图形化界面，并包含 parted 和 ntfsresize 功能。gtparted 工具是另一个基于 parted 的图形化分区工具，是为 GNOME 桌面设计的，它使用 GTK+GUI 库。
- mac-fdisk。这个工具主要用在 Power PC 版本的 Linux 中，常用的 X86 版本中没有这个工具。

还有几个商业分区工具，其中最有名的可能是 PartitionMagic，现在由 Symantec 销售。许多发行版允许在安装过程中对硬盘进行分区，有时候还允许缩小现有的 Windows NTFS 或 FAT32 分区，请参考发行版的安装文档。

3.31 mkinitrd：建立要载入 ramdisk 的映象文件

作用：建立要载入 ramdisk 的映象文件，以供 Linux 开机时载入 ramdisk。

用法：mkinitrd(选项)

主要选项如下。

- --builtin=<模块>：认为指定模块已经装入内核，忽略错误。
- -f：允许覆盖已存在的映象文件。
- --image-version：内核版本号将附加到建立的映象文件的目录前。
- --fstable=<文件系统列表>：使用列表自动探测根设备所建立的文件系统类型。
- --nocompress：不压缩生成的映象文件。
- --nopivot：不使用 pivot_root 系统调用作为映象的一部分。
- --omit-lvm-modules：不载入任何 LVM 模块。
- --omit-scsi-modules：不载入任何 SCSI 模块。
- --preload=<模块>：将指定的模块载入映象中。
- -v：在创建映象过程中打印信息。
- -version：打印程序版本信息。

应用实例如下。

在/tmp 目录下建立一个映象文件。

```
# mkinitrd -v -f myinitrd.img $(uname -r)
```

3.32 ssm：命令行集中存储管理工具

作用：system storage manager（ssm）是命令行存储管理工具。这个工具可以用于存储设备的集中管理。ssm 工具可以用来取代之前的许多管理命令，如 fdisk、btrfs、cryptsetup、lvm2、

mdadm、resize2fs 等。

用法：ssm [-h] [--version] [-v] [-f] [-b backend] [-n]{check,resize,create,list,add,remove,snapshot,mount} ...

主要选项如下。

- -f,--force：强制执行命令（笔者不推荐使用这个选项）。
- -bbackend：选择后端使用。目前可以选择（lvm，btrfs，crypt）。
- -n,--dry：模拟运行。这个选项主要用于调试的目的。
- -h,--help：显示帮助信息然后退出。
- --version：显示版本信息然后退出。

主要子命令如下。

- check：检查存储设备上的文件系统的一致性。
- resize：调整文件系统大小。
- create：建立一个新的卷并且定义参数。
- list：显示设备列表，包括存储池、卷、快照等。
- add：添加一个或者多个设备到存储池。
- remove：从存储池删除一个或者多个设备。
- snapshot：建立快照。
- mount：在指定位置挂载文件系统。

应用实例如下。

1）查看所有设备列表

执行如下命令查看所有设备列表：

```
# ssm list
```

结果如图 3-3 所示。

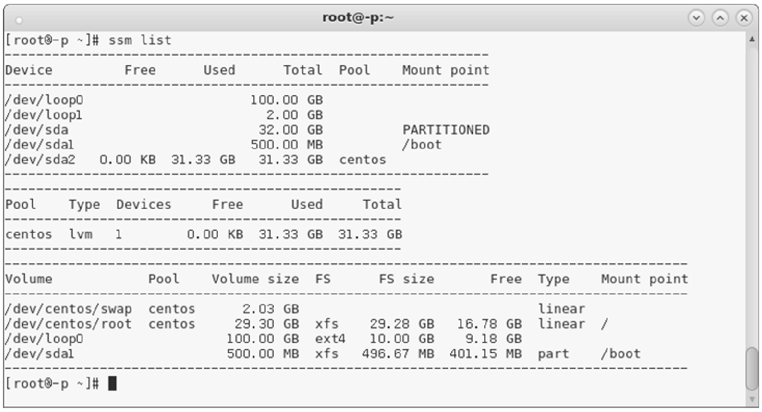


图 3-3　查看所有设备列表

图 3-3 显示的信息包括三大部分：首先显示的是物理存储设备（/dev/sda）信息，中间是存储池信息。最下面是 LVM 逻辑卷信息。

list 子命令支持如下参数。

- {volumes | vol}：系统中的所有卷信息列表。
- {devices | dev}：所有设备信息列表，包括一些存储设备，例如光盘 DM / MD 设备。
- {pools | pool}：系统中所有存储池信息列表。
- {filesystems | fs}：系统中所有文件系统信息列表。
- {snapshots | snap}：所有快照信息列表。

2）将物理磁盘添加到 LVM 池。

如果将一个新的物理磁盘（如/dev/sdb）添加到现有的存储池（如 centos），那么将新的物理磁盘添加到现有存储池的命令如下：

```
ssm add -p  < pool-name >   < device >
#  ssm add –p centos /dev/sdb (ENTER)
```

新设备添加到存储池后，存储池会自动扩大，扩大多少取决于设备的大小。

3）扩大 LVM 卷

如果在存储池中有额外空间，可以扩大存储池中现有的磁盘卷。可以使用 ssm 命令的 resize 选项。

```
ssm resize -s [size] [volume]
```

resize 子命令支持的存储单位包括：K 为千字节，M 为兆字节，G 为千兆字节，T 为 TB 级，默认值是千字节。

例如，将/dev/centos/root 卷的大小增加 50MB。

```
# ssm resize -s+50M /dev/centos/root  (ENTER)
```

4）查看、检查存储池和文件系统

可以使用如下命令查看存储池列表：

```
# ssm list pool (ENTER)
-----------------------------------------------------
Pool     Type  Devices     Free     Used    Total
-----------------------------------------------------
centos    lvm   2       36.00 MB  31.33 GB  31.36 GB
lvm_pool  lvm   1       36.00 MB   0.00 KB  36.00 MB
```

可以使用如下命令检查存储池和文件系统：

```
# ssm check lvm_pool/crash_vol (ENTER)
Checking ext4 file system on '/dev/mapper/lvm_pool-crash_vol':
Device '/dev/mapper/lvm_pool-crash_vol' is mounted on '/crash' Unmount
(N/y/q) ? Y
e2fsck 1.42.5 (29-Jul-2014)
...
```

```
/dev/mapper/lvm_pool-crash_vol: 12/655360 Dateien (0.0% nicht
zusammenhangend), 61231/2621440 Blocke
OK
```

3.33 使用 xfs 管理命令

在 GNU/Linux 中，管理 XFS 的工作主要使用 xfsprogs 中的一系列工具。

1．命令简介

- mkfs.xfs：创建 XFS 文件系统。
- xfs_admin：调整 XFS 文件系统的各项参数。
- xfs_copy：复制 XFS 文件系统的内容到一个或多个目标系统（并行方式）。
- xfs_db：调试或检测 XFS 文件系统（查看文件系统碎片等）。
- xfs_check：检测 XFS 文件系统的完整性。
- xfs_bmap：查看一个文件的块映射。
- xfs_repair：尝试修复受损的 XFS 文件系统。
- xfs_fsr：碎片整理。
- xfs_quota：管理 XFS 文件系统的磁盘配额。
- xfs_metadump：将 XFS 文件系统的元数据（Metadata）复制到一个文件中。
- xfs_mdrestore：从一个文件中将元数据（Metadata）恢复到 XFS 文件系统。
- xfs_growfs：调整 XFS 文件系统的大小（只能扩展）。
- xfs_freeze：暂停（-f）和恢复（-u）XFS 文件系统。

2．建立 XFS 文件系统

1）格式化

要格式化存储设备为 XFS 格式，可以用 root 身份执行如下命令：

```
#mkfs -t xfs /dev/sdb5
```

如果 mkfs.xfs 发现存储设备仍有之前存放的文件资料，则会拒绝进行格式化。

```
# mkfs.xfs /dev/sdb5
mkfs.xfs: /dev/sdb5 appears to contain an existing filesystem (xfs).
mkfs.xfs: Use the -f option to force overwrite.
```

如果确定那些资料已没有用处，需要为 mkfs.xfs 加上选项 -f 强迫它进行格式化。

```
# mkfs.xfs -f /dev/sdb5
meta-data   =/dev/sdb5     isize=256  agcount=4,agsize=524119 blks
            =              sectsz=512 attr=2
data        =              bsize=4096        blocks=2096474,imaxpct=25
```

```
         =              sunit=0s          width=0 blks
naming   =version 2     bsize=4096
log      =internal log  bsize=4096        blocks=2560, version=2
         =              sectsz=512        sunit=0 blks, lazy-count=0
realtime =none          extsz=4096        blocks=0, rtextents=0
```

2）区块大小（Block size）

区块（Block）是文件系统存储盘中内容最小的单位，其大小对文件系统的空间运用和效用有很大的影响。较大的区块可以令文件系统大小上限和文件大小上限增加，也可以加快大文件的读/写，但会浪费较多的空间，对平均文件大小较小的文件系统比较不利。区块大小只可以在格式化文件系统时设定，以后除重新格式化外不能改变。XFS 文件系统的区块大小最少可以为 512 字节，最大不可超过 64KB，默认为 4KB。然而区块大小又受到作业系统内核的 page 大小限制。在 X86 计算机中，区块最大不可超过 4KB。其他平台如 IA64 可以使用较大区块，不过过大区块会浪费空间，所以不建议使用大于 4KB 的区块。在选择区块大小时要注意以下几点：

- 如果文件系统小于 100MB 或有大量小型文件，建议使用 512 字节区块，其余情况建议使用 4KB 区块。
- 如果用作新闻组服务器（News Server）或有大量小型文件，可以使用 512 字节文件系统区块和 4KB 目录区块（使用-nsize=大小选项）。

简单而言，XFS 文件系统在 X86 平台上可以使用 512B、1KB、2KB 和 4KB 区块。格式化显要指定区块大小需要使用选项“-b size=区块大小”：

```
#mkfs.xfs –b size=512 /dev/sdb6
```

3）目录区块大小（Directory Block Size）

XFS 容许目录使用比文件系统区块大小更小的区块，方法是使用选项“-n size=区块大小”，例如：

```
#mkfs.xfs –b size=512 –n size=4k /dev/sdb6
```

区块大小后加上“k”表示单位为 KB（1024 字节），加上“s”表示单位为磁区（sector，默认为 512 字节，可能会因-s 选项而改变），加上“b”表示单位为文件系统区块（默认为 4KB，可能会因-b 选项而改变）。

4）日志大小

格式化 XFS 时，mkfs.xfs 会自动根据文件系统的大小划分日志（Journal）的大小。若文件系统等于或超过 1TB，则划分日志只会为最大值 128MB。最小不会小于 512 文件系统区块。可以使用选项“-l size=日志大小”指定日志的大小，例如：

```
#mkfs.xfs –l size=1024b /dev/sdb6
```

日志大小可以加以下单位。

- s：磁区（sector）大小（默认为 512 字节，可能会因-s 选项而改变）。
- b：文件系统区块大小（默认为 4KB，可能会因-b 选项而改变）。

– k：KB（1 024 字节）。
– m：MB（1 048 576 字节）。
– g：GB（1 073 741 824 字节）。
– t：TB（1 099 511 627 776 字节）。
– p：PB（1024TB）。
– e：EB（1 048 576TB）。

如果有多于一个硬盘，可以考虑使用外部日志（External Journal）把文件系统和日志存储在不同的硬盘，可以增加效能。

5）文件系统标签（Filesystem Label）

文件系统标签（Filesystem Label）又叫作 Volume Name，是文件系统中一个小栏目，用作简述该文件系统的用途或其存储数据。可以使用选项“-L 标签”在格式化时设定文件系统标签。

```
#mkfs.xfs -L Videos /dev/sdc1
```

XFS 的文件系统标签不能超过 12 个字符。以后可以使用命令 xfs_admin -L 改变。

6）一个例子

示例代码如下：

```
#mkfs.xfs –d agcount=4 –l size=32m /dev/sdb5
```

第一个选项是-l size=32m，它告诉 mkfs.xfs 配置用户的文件系统使之拥有一个高达 32MB 的元数据日志。这通过降低在文件系统处于繁忙使用期间元数据日志将“填满”的可能性而改善了性能。第二个选项通过告诉 mkfs.xfs 将创建的分配组的数目最小化，让用户增强新文件系统的性能。通常，mkfs.xfs 自动选择分配组的数目，但是，根据笔者的经验，它通常会选择一个比大多数用于一般用途的 Linux 工作站和服务器过高一点的数目。分配组让 XFS 并行执行多个元数据操作，这为高端服务器带来了便利，但是太多的分配组确实会增加一些开销。因此，不要让 mkfs.xfs 为用户的文件系统选择分配组的数目，而是通过使用-d agcount=x 选项指定一个数目。将 x 设置成一个小数目，如 4、6 或 8。需要使得目标块设备中每 4GB 容量至少有一个分配组。同时进行这两项调整，使用下面的命令创建“优化的”XFS 文件系统：

```
# mount -t /dev/sda6   /mnt -o noatime,nodiratime,osyncisdsync
```

前面的两个选项（noatime,nodiratime）关闭 atime 更新。osyncisdsync 选项调整 XFS 的同步 / 异步行为，以便它同 Ext3 更一致。

3．挂载 XFS 文件系统

```
#mount –t xfs/dev/sdb5/xfs
```

其中，/xfs 是主分区/下的一个目录。

为了让系统启动后就自动加载，应该更改/etc/fstab，这样系统启动后就会自动加载 xfs 分区而不必每次都手工加载。添加如下一行：

```
/dev/hdb5 /xfs defaults 1 1
```

挂装时，将使用一些性能增强 mount 选项来最大限度地发掘出（或发挥出）新文件系统的性能。

```
#mount –t /dev/sdb5 /xfs –o noatime,nodiratime,osyncisdsync
```

前面的两个 mount 选项关闭 atime 更新，几乎不需要 atime 更新，并且它除了降低文件系统性能之外几乎不起任何作用。osyncisdsync 选项调整 XFS 的同步 / 异步行为，以便它同 Ext3 更一致。多亏了 mkfs.xfs 和 mount 调整，新的 XFS 文件系统比没有调整时的性能要好得多。

其他 mount -o 选项如下。

- allocsize=：延时分配时，预分配 buffered 大小。
- sunit=/swidth=：使用指定的条带单元与宽度（单位为 512Byte）。
- swalloc：根据条带宽度的边界调整数据分配。
- discard：块设备自动回收空间。
- dmapi：使能 Data Management API 事件。
- inode64：创建 inode 节点位置不受限制。
- inode32：inode 节点号不超过 32 位（为了兼容）。
- largeio：大块分配。
- nolargeio：尽量小块分配。
- noalign：数据分配时不用条带大小对齐。
- noatime：读取文件时不更新访问时间。
- norecovery：挂载时不运行日志恢复（只读挂载）。
- logbufs=：内存中的日志缓存区数量。
- logbsize=：内存中每个日志缓存区的大小。
- logdev=/rtdev=：指定日志设备或实时设备。XFS 文件系统可以分为 3 部分：数据、日志、实时（可选）。

4. 调整 XFS 文件系统各项参数

1）XFS 卷标管理

（1）查看当前的卷标。

```
#xfs_admin -l /dev/sdb
label=""
```

（2）设置新的卷标。

```
#xfs_admin -L "VideoRecords" /dev/sdb
writing all SBs
new label="VideoRecords"
```

2）UIID 管理

通用唯一标识符（UUID）是 128 比特的数字，用来唯一地标识因特网上的某些对象或者实体。传统上，GNU/Linux 在/etc/fstab 上直接使用设备名称（/dev/hda1 或/dev/sda5 等）指定要挂载的存储设备。然而设备名称有时会因为 BIOS 的设定而改变，引起混乱。所以现在部分

Linux distribution 已改用 UUID（Universal Unique Identifier）来指定要挂载的存储设备。

（1）查看当前所有存储设备的 UUID 名称。

```
#blkid -s UUID
/dev/sda1:UUID="34dd521d-fb74-41cf-afc6-e786344ecd7a"
/dev/sda2:UUID="UskH3q-GHDB-ZLoo-kPRb-O1sq-wKSU-CwH0Lt"
/dev/mapper/rhel-root:UUID="e7e811fd-3c45-4bcd-84cb-92c4aafccf16"
/dev/sdb:UUID="36cf1092-65e2-4acd-85fc-284b1e7b1f33"
/dev/mapper/rhel-swap:UUID="800748d6-f4ae-4bc7-90d9-e69478fd4af3"
```

（2）查看指定存储设备的 UUID。

```
#xfs_admin -u /dev/sdb
UUID=cd4f1cc4-15d8-45f7-afa4-2ae87d1db2ed
```

（3）生成一个新的 UUID。

```
#xfs_admin -U generate /dev/sdb
writing all SBs
new UUID=c1b9d5a2-f162-11cf-9ece-0020afc76f16
```

-U 的参数如果为 generate，则表示直接产生一个新的 UUID；如果为 nil，则表示清除文件系统的 UUID。

```
#xfs_admin -U nil /dev/sda1
```

3）在 mount 命令中使用 UUID 挂载文件系统

使用 mount 命令挂载文件系统，可以使用选项“-U uuid”取代设备文件指定要挂载的设备。

```
mount -U 51f7e9a4-5154-4e29-a7a6-208417290b85 /mnt
```

也可以使用 UUID=uuid 取代-U 选项。

```
mount UUID="51f7e9a4-5154-4e29-a7a6-208417290b85" /mnt
```

在文件/etc/fstab 中可以使用 UUID=uuid 取代设备文件指定要挂载的设备。

```
UUID="e61f4197-5f00-4f4f-917c-290922a85339" /xfs defaults 0 1
UUID="51f7e9a4-5154-4e29-a7a6-208417290b85" /boot xfs defaults 0 2
```

5．在线调整 XFS 文件系统的大小

XFS 提供了 xfs_growfs 工具，可以在线调整 XFS 文件系统的大小。XFS 文件系统可以向保存当前文件系统的设备上的未分配空间延伸。这个特性常与卷管理功能结合使用，因为后者可以把多个设备合并进一个逻辑卷组，而使用硬盘分区保存 XFS 文件系统时，每个分区需要分别扩容。

```
# xfs_growfs -D 1073741824 /myxfs1
# xfs_growfs -d /myxfs1
```

6. 暂停和恢复 XFS 文件系统

（1）暂停 XFS 文件系统。

```
#xfs_freeze –f /myxfs
```

（2）恢复 XFS 文件系统。

```
#xfs_freeze –u /myxfs
```

7. 尝试修复受损的 XFS 文件系统

XFS 与 Ext3 相比的特点是并行 I/O，如果一个文件系统使用的硬盘比较多，而且总线允许并行的话，XFS 有明显的性能优势。而在台式计算机上，这个区别很不明显。

另外，Ext3 删除文件的速度比 XFS 要快；由于大量采用 Cache，XFS 不用 fsck，但必须保证电源供应，突然断电时 XFS 的损失比 Ext3 要严重。

```
# # xfs_repair –L /dev/hda13
```

8. 备份和恢复

（1）备份文件系统。

```
#xfsdump-F –f /root/dump.xfs /mnt
```

（2）恢复文件系统。

```
#xfsrestore –f /root/dump.xfs /mnt
```

9. 碎片管理

可以使用 xfs_db 命令调试或检测 XFS 文件系统（查看文件系统碎片等）。

（1）查看碎片情况。

```
#xfs_db –c frag –r /dev/sda1
actual 378,idea l373,fragmentation factor 1.32%
```

（2）整理碎片。

```
#xfs_fsr /dev/sda1
```

3.34 LVM 命令列表

LVM（Logical Volume Manager，逻辑卷管理器）是一种把硬盘驱动器空间分配成逻辑卷的方法，使硬盘不必使用分区也能被简单地重新划分大小。LVM 是 Linux 环境下对磁盘分区进行管理的一种机制，它将多个物理分区汇聚为一个卷组（Volume Group），而且这些物理卷的大小可以不同，甚至类型也可以不同（如 SCSI 和 SATA 磁盘）。组成的卷组就像一块大硬盘，然后从中分割出一块一块的逻辑卷（Logical Volume），并进一步在逻辑卷组上创建文件系统。LVM 有一整套的命令集，表 3-2 列出了 LVM 命令摘要。

表 3-2　LVM 命令摘要和示例

	命令名称	功能描述	主要参数
物理卷（PV）管理命令	pvcreate	将物理硬盘分区初始化为物理卷，以便被 LVM 使用	-f：强制创建物理卷，不需要用户确认 -u：指定设备的 UUID -y：所有的问题都回答“yes” -Z：是否利用前 4 个扇区
	pvdisplay	显示物理卷的属性（大小，PE 大小，被卷组描述符使用的空间）	-s：以短格式输出 -m：显示 PE 到 LV 和 LE 的映射
	pvchange	设置 PV 的性能，允许或拒绝从这个磁盘上分配另外的 PE	-u：生成新的 UUID -x：是否允许分配 PE
	pvmove	在一个卷组中从源到目的移动已经分配的 PE	-d：调试模式 -f：强制删除 -y：对提问回答“yes”
	pvck	检查一致性	-d：调试模式
	pvs	输出格式化的物理卷信息报表。使用 pvs 命令仅能得到物理卷的概要信息，如果要得到更加详细的信息，可以使用 pvdisplay 命令	--noheadings：不输出标题头 --nosuffix：不输出空间大小的单位
	pvscan	扫描系统中连接的所有硬盘，列出找到的物理卷列表	-d：调试模式 -e：仅显示属于输出卷组的物理卷 -n：仅显示不属于任何卷组的物理卷 -s：短格式输出 -u：显示 UUID
卷组（VG）管理命令	vgcreate	创建卷组	-l：卷组中允许创建的最大逻辑卷数 -p：卷组中允许添加的最大物理卷数 -s：卷组中的物理卷的 PE 大小
	vgdisplay	显示 LVM 卷组的元数据信息	-A：仅显示活动卷组的属性 -s：使用短格式输出信息
	vgchange	修改卷组的属性，经常被用来设置卷组是处于活动状态还是非活动状态	-a：设置卷组的活动状态
	vgextend	用于动态扩展 LVM 卷组，它通过向卷组中添加物理卷来增加卷组的容量	-d：调试模式 -t：仅测试
	vgscan	查找系统中存在的 LVM 卷组，并显示找到的卷组列表	-d：调试模式 --ignorelockingfailure：忽略锁定失败的错误
	vgreduce	通过删除 LVM 卷组中的物理卷来减少卷组容量。不能删除 LVM 卷组中剩余的最后一个物理卷	-a：如果命令行中没有指定要删除的物理卷，则删除所有的空物理卷 --removemissing：删除卷组中丢失的物理卷，使卷组恢复正常状态

续表

	命令名称	功能描述	主要参数
	vgremove	删除卷组	-f：强制删除
	vgexport	从系统中删除一个卷组，但不修改基于物理卷的信息	-a,--all：导出所有非活动卷组
	vgcfgrestore	恢复卷组的配置信息	--force：强制恢复 LVM：配置数据 -l：列出在指定的配置备份文件中保存的配置信息 -f,--file：filename：从指定的配置备份文件中获取配置信息
	vgimport	通过扫描使用 vgexport 命令输出物理卷在系统中增加的一个卷组	-d：调试模式 -a,--all：导入所有卷组
	vgcfgbackup	保存卷组的配置信息，记住一个卷组由一个或多个物理卷组成	-d：调试模式
	vgck	检查一致性	-d：调试模式
	vgsplit	把一个卷组拆分为两个	-A：autobackup 此次调用设置自动备份 -d：调试模式
	vgmknodes	重新建立已有卷组的卷组目录和其中的设备文件	--refresh：如果卷组中的任何逻辑卷处于活动状态，则重新载入它的元数据
逻辑卷（LV）管理命令	lvcreate	创建逻辑卷	-L：指定逻辑卷的大小，单位为“kKmMgGtT”字节 -l：指定逻辑卷的大小（LE 数） -n：后面跟逻辑卷名
	lvdisplay	显示逻辑卷的信息	-m,--maps：显示逻辑盘区物理卷和物理盘区的映射 --columns,-C：显示输出列 -c,--colon：生成冒号分隔输出的脚本
	lvchange	改变逻辑卷的特性：可用性、调度策略、权限、块重定位、分配策略、镜像缓存的可用性	-A autobackup：此次调用设置自动备份 -C contiguous：设置连续分配策略
	lvextend	增加逻辑卷的空间	-L：指定逻辑卷的大小，单位为“kKmMgGtT”字节 -l：指定逻辑卷的大小（LE 数）
	lvreduce	减少逻辑卷的空间	-L：指定逻辑卷的大小，单位为“kKmMgGtT”字节 -l：指定逻辑卷的大小（LE 数）
	lvremove	删除逻辑卷	-f：强制删除

续表

	命令名称	功能描述	主要参数
	lvrename	修改逻辑卷名称	--noudevsync：禁用 udev 的同步 -d：调试模式 -t：仅测试
	lvmdiskscan	检测所有的 SCSI、IDE 等存储设备，并输出摘要信息，包括名称、大小、类型等	-l,--lvmpartition：只报告物理卷
	lvscan	扫描当前系统中存在的所有 LVM 逻辑卷	-b：显示逻辑卷的主设备号和次设备号
	lvmsar	收集的逻辑卷的读/写统计数据	-d：调试模式

应用实例如下。

扩展 LVM 需要执行如下步骤。

若卷组中无剩余空间，首先扩展卷组。

- 添加硬盘。
- 在硬盘上创建 LVM 分区。
- 在 LVM 分区上创建物理卷。
- 将物理卷添加到卷组中。

若卷组中有剩余空间，则扩展卷组中的逻辑卷。

对已扩展的逻辑卷中的文件系统要进行容量扩展。

以下是使用 LVM 安装 RHEL 5.4 后扩展 LogVolHome 逻辑卷的操作过程。

- 在系统中添加一块新硬盘。
- 在新硬盘上创建物理卷。
- 将新创建的物理卷扩展到 VolGroup00 卷组。
- 扩展 VolGroup00 卷组中的 LogVolHome 逻辑卷。
- 对 LogVolHome 逻辑卷上的文件系统进行容量扩展。

1）创建 LVM 类型的分区

可以用前面介绍的 fdsik 进行分区，也可以用 parted 命令。下面给出一个使用 parted 命令进行分区的过程示例。

```
# parted /dev/sdc mktable msdos
Information: Don't forget to update /etc/fstab, if necessary.
# parted /dev/sdc mkpart primary 0 4GB
# parted /dev/sdc mkpart extended 4GB 8GB
# parted /dev/sdc mkpart logical 4GB 8GB
# parted /dev/sdc set 1 lvm on
Information: Don't forget to update /etc/fstab, if necessary.
# parted /dev/sdc set 5 lvm on
Information: Don't forget to update /etc/fstab, if necessary.
```

```
# parted /dev/sdc print
Model: VMware, VMware Virtual S (scsi)
Disk /dev/sdc: 17.2GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Number Start End Size Type File system Flags
1 32.3kB 3997MB 3997MB primary lvm
2 3997MB 8003MB 4006MB extended lba
5 3998MB 8003MB 4006MB logical lvm
Information: Don't forget to update /etc/fstab, if necessary.
# partprobe
```

2）创建物理卷

– 扩展卷组。

扩展卷组需要使用 vgextend 命令，其命令格式如下：

```
vgextend <卷组名> <物理卷设备名> [<物理卷设备名> ...]
```

首先查看系统中的卷组。

```
# vgscan
Reading all physical volumes. This may take a while...
Found volume group "wwwVG" using metadata type lvm2
Found volume group "VolGroup00" using metadata type lvm2
```

下面的命令将新创建的两个物理卷扩展到已存在的 VolGroup00 卷组中。

```
# vgextend VolGroup00 /dev/sdc1 /dev/sdc5
Volume group "VolGroup00" successfully extended
```

扩展后查看卷组状态。

```
# vgdisplay
…
```

– 扩展逻辑卷。

要扩展卷组中的逻辑卷，需要使用 lvextend 命令，其命令格式如下：

```
lvextend <-L/--size+逻辑卷增量[kKmMgGtT]><逻辑卷设备名称>
```

在创建逻辑卷时直接指定增量，单位可以是 k/K、m/M、g/G、t/T，默认为 M。

首先显示已存在的 LogVolHome 逻辑卷的状态。

```
# lvdisplay /dev/VolGroup00/LogVolHome
--- Logical volume ---
LV Name /dev/VolGroup00/LogVolHome
VG Name VolGroup00
```

```
LV UUID q6j2U1-g5u9-inAj-zWNy-hrK7-T71j-N59hMZ
LV Write Access read/write
LV Status available
# open 1
LV Size 2.91 GB
Current LE 93
Segments 1
Allocation inherit
Read ahead sectors 0
Block device 253:1
```

下面的命令用于在 VolGroup00 卷组中扩展 LogVolHome 逻辑卷，扩展大小为 6GB。

```
# lvextend -L +6G /dev/VolGroup00/LogVolHome
Extending logical volume LogVolHome to 8.91 GB
Logical volume LogVolHome successfully resized
```

重新显示 LogVolHome 逻辑卷的状态。

```
# lvdisplay /dev/VolGroup00/LogVolHome
--- Logical volume ---
LV Name /dev/VolGroup00/LogVolHome
…
```

– 扩展文件系统容量。

下面是对已挂装的 Ext3 类型的 home 文件系统扩充容量的操作。

```
# df /home
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVolHome
2951952 107668 2691916 4% /home
# umount /home
# resize2fs -f /dev/VolGroup00/LogVolHome
resize2fs 1.39 (29-May-2006)
Resizing the filesystem on /dev/VolGroup00/LogVolHome to 2334720 (4k) blocks.
The filesystem on /dev/VolGroup00/LogVolHome is now 2334720 blocks long.
# mount -a
# df /home
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVolHome
9052536 109888 8569096 2% /home
```

对文件系统调整大小是非常危险的操作，虽然从技术上考虑是可行的，但是仍然有很大风险，因此，用户在使用 resize2fs 命令调整文件系统容量之前必须进行完整备份。使用 resize2fs 命令之前需卸载文件系统，若要对/分区、/usr 分区等进行操作，需要使用 Live-CD 的 rescue 模式启动系统。

04 Linux 进程管理命令

4.1 accton：打开或关闭进程统计

作用：打开进程统计。如果不带任何参数，即关闭进程统计。

用法：accton[-V|--version] [-h|--help] [filename]

主要选项如下。

- -V,--version：显示 ac 版本并退出。
- -h,--help：打印命令概要，并显示系统默认的 accton 文件。

> **说明**
>
> acct 是一个工具包，里面包含针对用户连接时间、进程执行情况等进行统计的工具。它可以记录用户登录信息、用户所执行的程序、程序执行情况等信息。它包含下面一些命令。

- ac：登录账号的简要信息。
- accton：打开或关闭进程账号记录功能。
- last：显示曾经登录过的用户。
- lastcomm：显示已执行过的命令。
- sa：进程账号记录信息的摘要。
- dump-utmp：输出 utmp 文件的内容。
- dump-acct：输出 acct 或 pacct 文件的内容。

应用实例如下

打开或关闭进程统计。

在 Linux 系统中启动进程统计使用 accton 命令，必须用 root 身份来运行。accton 命令的形式为 accton file，file 必须事先存在。首先使用 touch 命令创建 pacct 文件。

```
#touch /var/log/pacct
```

然后运行如下程序。

```
# accton /var/log/pacct
```

一旦 accton 被激活，就可以使用 lastcomm 命令监测系统中任何时候执行的命令。若要关闭统计，可以使用不带任何参数的 accton 命令。

```
# accton
```

4.2 lastcomm：显示以前使用过的命令的信息

作用：accton 命令用来启动进程记录，这样就会把所有的命令都记录到一个指定的文件中，而 lastcomm 命令就是用来查看这个文件的，以方便系统管理。

用法：lastcomm [选项]...[文件]...

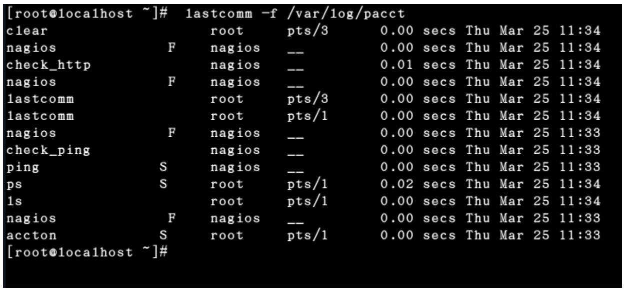
主要选项如下。

- -strict-match：精确匹配每一列。
- --user name：只显示指定用户的命令记录。
- --command name：只显示指定命令的记录。
- --tty name：只显示在指定终端上运行的命令。
- -f filename：指定一个命令记录文件来代替默认文件——acct。
- --debug：打印其他内核信息。
- -V,--version：打印版本。
- -h,--help：打印概要和系统默认统计文件（Linux 里面的默认文件多是/var/log/pacct 和/var/account/pacct）。

应用实例如下。

显示曾经执行过的命令，如图 4-1 所示。

```
# lastcomm -f /var/log/pacct
```



```
[root@localhost ~]#  lastcomm -f /var/log/pacct
clear                   root     pts/3      0.00 secs Thu Mar 25 11:34
nagios            F     nagios   __         0.00 secs Thu Mar 25 11:34
check_http              nagios   __         0.01 secs Thu Mar 25 11:34
nagios            F     nagios   __         0.00 secs Thu Mar 25 11:34
lastcomm                root     pts/3      0.00 secs Thu Mar 25 11:34
lastcomm                root     pts/1      0.00 secs Thu Mar 25 11:34
nagios            F     nagios   __         0.00 secs Thu Mar 25 11:33
check_ping              nagios   __         0.00 secs Thu Mar 25 11:33
ping             S      nagios   __         0.00 secs Thu Mar 25 11:33
ps               S      root     pts/1      0.02 secs Thu Mar 25 11:34
ls                      root     pts/1      0.00 secs Thu Mar 25 11:34
nagios            F     nagios   __         0.00 secs Thu Mar 25 11:33
accton           S      root     pts/1      0.00 secs Thu Mar 25 11:33
[root@localhost ~]#
```

图 4-1　显示曾经执行过的命令

每一项均包含如下信息。

- 进程的命令。
- 标志，由系统标准统计进程完成。

Ø S：命令由超级用户运行。

Ø F：命令由子进程运行，没有使用 exec 的子进程。

Ø C：命令运行在 PDP-11 兼容模式下。

Ø D：命令终止时产生 core 文件。

Ø X：命令由信号 SIGTERM 终止。

- 运行命令的用户名。
- 进程使用的系统时间。

4.3 sa：报告、清理并维护进程统计文件

作用：sa 命令把/var/log/pacct 文件中的信息压缩到摘要文件/var/log/savacct 和/var/log/usracct 中，报告、清理并维护进程统计文件。

用法：sa [选项]...[文件]...

主要选项如下。

- -a：打印所有命令的名称（包括那些带有不可打印字符的）。
- -b：将输出按用户和系统时间的总和除以调用次数来进行分类；否则，输出为用户时间和系统时间之和。
- -c：将每个命令的使用时间打印为相对于所有命令使用时间的百分比。此外，还有用户、系统和实时。
- -C：将记账文件合并到摘要文件中。如果摘要文件为旧用法，则将它转换为新用法。
- -d：将输出按平均磁盘 I/O 操作数进行分类。
- -D：将输出按总的磁盘 I/O 操作数进行分类和打印。
- -f：不要强制进行交互式阈值压缩。此标志必须与-v 标志一起使用。
- -I：仅读取原始数据，不读取摘要文件。
- -j：打印每个调用的秒数，而不是每个类别的总的分钟数。
- -k：将输出按平均 CPU 时间进行分类。
- -K：将输出按 CPU 存储量整数进行分类和打印。
- -l：将系统时间和用户时间分离，而不是组合它们。
- -m：打印每个用户的进程数和 CPU 分钟数。
- -n：按调用数对输出进行分类。
- -r：将分类的顺序倒置。
- -s：将记账文件合并到摘要文件中。
- -t：打印每个命令的用时与用户和系统时间和之比。
- -u：暂挂所有其他标志并且打印每个命令的用户数字标识和命令名。

与 sa 命令一起使用的标志改变报告信息的类型。报告可以包含以下字段。

- avio：每次执行的平均 I/O 操作次数。

- cp：用户和系统时间总和，以分钟计。
- cpu：和 cp 一样。
- k：内核使用的平均 CPU 时间，以 k 为单位。
- k*sec：CPU 存储完整性，以 1k-core 秒为单位。
- re：实时时间，以分钟计。
- s：系统时间，以分钟计。
- tio：I/O 操作的总数。
- u：用户时间，以分钟计。

如果不指定任何标志而运行 sa 命令，摘要报告将包含每个命令的调用次数，以及 re、cpu、avio 和 k 字段。

应用实例如下。

（1）打印每个用户的进程数和 CPU 分钟数。

```
# sa -m /var/log/pacct
                           904     72.81re      0.27cp     1324k
root                       658     69.54re      0.27cp     926k
nagios                     237     3.24re       0.00cp     2400k
sshd                         8      0.00re       0.00cp     2104k
smmsp                        1      0.02re       0.00cp     2035k
```

（2）指定 sa 读取手工建立的文件/var/account/acct。

使用 sa 命令默认读取/var/account/下的 pacct、savacct 和 usracct 文件，我们需要指定它读取手工建立的文件/var/account/acct。

```
# sa -a --other-acct-file  /var/account/acct
      3      0.00re      0.00cp        0avio      726k
      1      0.00re      0.00cp        0avio     1207k   ls
      1      0.00re      0.00cp        0avio      571k   goodc
      1      0.00re      0.00cp        0avio      399k   accton
```

（3）按平均 CPU 时间进行记账记录总结。

```
sa  -k
```

4.4 at：定时运行命令

作用：在指定时刻执行指定的命令序列。

用法：at [-V] [-q 队列] [-f file] [-m] time

主要选项如下。

- -V：输出版本号。

– -q：选择队列名称，队列名称可以是 a～z 和 A～Z 之间的任意字母。队列字母顺序越高，则队列优先级别越低。
– -f：从文件中读取命令或 Shell 脚本，而非在提示后指定它们。
– -m：执行完作业后发送电子邮件给用户。
– time：设定作业执行的时间。time 参数可以是下面用法中的任何一种。
  Ø HH:MM 用法——如 04:00，代表 4:00AM。如果时间已过，就会在第二天的这一时间执行。midnight 代表 12:00AM。noon 代表 12:00PM。teatime 代表 4:00PM。
  Ø 英文月名、日期、年份用法——如 January 15 2005，代表 2005 年 1 月 15 日。年份可无。
  Ø MMDDYY、MM/DD/YY 或 MM.DD.YY 用法——如 011505，代表 2005 年 1 月 15 日。
  Ø now +时间用法——时间以 minutes、hours、days 或 weeks 为单位。如 now + 5 days，代表命令应该在 5 天之后的此时此刻执行。偏移量的用法为时间＋偏移量，单位是 minutes、hours 和 days。

说明

at 命令实际上是一组命令集合。at 命令在指定时刻执行指定的命令序列。atq 查询已经设定的命令序列。atrm 删除已经设定的命令序列。batch 当系统负载低于一个设定值（0.8）时执行设定的命令序列。

应用实例如下。

让系统在两天后的 15:30 执行文件 data 中指定的作业。

```
# at -f date 15:30 +2 days
job 3 at 2010-03-27 15:30
```

4.5 atq：显示目前使用 at 命令后待执行的命令队列

作用：查询已经设定的命令序列。在用户是超级用户的情况下，列出所有用户的作业。

用法：atq [-V] [-q 队列]

主要选项如下。

– -V：输出版本号。
– -q：选择队列名称，队列名称可以是 a～z 和 A～Z 之间的任意字母。队列字母顺序越高，则队列优先级别越低。

应用实例如下。

显示目前使用 at 命令后待执行的命令队列。

```
# atq
3       2010-03-27 15:30 a root
2       2010-03-26 15:30 a root
```

4.6 atrm：删除 at 命令中待执行的命令队列

作用：删除 at 命令中待执行的命令队列。

用法：atrm [-V]作业号

主要选项如下。

-V：输出版本号。

应用实例如下。

删除 at 命令中待执行的 3 号作业。

```
#atrm 3
```

4.7 batch：在系统负载水平允许的时候执行命令

作用：在系统负载水平允许的时候执行命令。换句话说，当平均负载低于 0.8，或降到了在 atrun 文件中指定的期望值时运行。

用法：batch [-V] [-q 队列] [-f 文件] [-mv] [time]

主要选项如下。

- -V：输出版本号。
- -q：选择队列名称，队列名称可以是 a～z 和 A～Z 之间的任意字母。队列字母顺序越高，则队列优先级别越低。
- -f：从文件中读取命令或 Shell 脚本，而非在提示后指定它们。
- -m：执行完作业后发送电子邮件给用户。
- -v：显示作业将要执行的时间。

应用实例如下。

1）配置 batch 作业

要在系统平均负载降到 0.8 以下时执行某项一次性的任务时，可以使用 batch 命令。输入 batch 命令后，“at>”提示就会出现。输入要执行的命令，按 Enter 键，然后输入“Ctrl-D”。可以指定多条命令，方法是输入每一条命令后按 Enter 键。输入所有命令后，按 Enter 键转入一个空行，然后再输入“Ctrl-D”。也可以在提示后输入 Shell 脚本，在脚本的每一行后按 Enter 键，然后在空行处输入“Ctrl-D”来退出。系统平均负载一旦降到 0.8，这组命令或脚本就会被执行。如果这组命令或脚本试图在标准输出中显示信息，则该输出会用电子邮件的方式被邮寄给用户。

2）进阶应用：控制对 at 和 batch 的使用

通过/etc/at.allow 和/etc/at.deny 文件可以限制对 at 和 batch 命令的使用。这两个使用控制文件的用法都是每行一个用户。两个文件都不允许使用空白字符。如果控制文件被修改了，at 守护进程不必被重启。每次用户试图执行 at 或 batch 命令时，使用控制文件都会被读取。不论控制文件如何规定，超级权限用户总是可以执行 at 和 batch 命令。如果 at.allow 文件存在，只有其中列出的用户才能使用 at 或 batch 命令，at.deny 文件会被忽略。如果 at.allow 文件不存在，所有在 at.deny 文件中列出的用户都将被禁止使用 at 和 batch 命令。

4.8 bg：后台运行命令

作用：使一个被挂起的进程在后台执行。

用法：bg

该命令无参数。

说明

Linux 作为一个多任务环境，用户会同时执行多项任务，例如，查看系统情况、备份资料、编辑文件和打印文件等。耗时长的任务不应该在前台任务中执行，而应该交给后台任务去执行，这样前台任务可继续正常进行其他的操作，不用等待。

应用实例如下。

（1）将一个大目录的文件进行排序后，将结果输入到一个文件中。

```
#du -a / | sort -rn > /tmp/du.sorted
```

（2）在命令后使用&符号。由于这个命令执行时间很长，可以把它放在后台运行。使用组合键 Ctrl+Z，然后使用 bg 命令。如果想直接把这个命令放在后台执行，可以在命令后使用“&”符号。

```
#du -a / | sort -rn > /tmp/du.sorted &
[1] 237
```

4.9 fg：挂起程序

作用：使一个被挂起的进程在前台执行。

用法：fg [job-spec]

主要选项如下。

[job-spec]：后台任务号码。

说明

fg 命令和 bg 命令是相对应的。如果想查看后台程序运行情况，可以使用 fg 命令把它调回前台查看。bg 命令可以将多个进程放到后台中执行。

应用实例如下。

使用 fg 命令时，要加入后台任务号码。如果不加任何号码，则所变动的均是当前任务。

```
#du -a / | sort -rn > /tmp/du.sorted &
[1] 237
#fg 1
```

4.10 jobs：显示后台程序

作用：显示后台任务的执行情况。

用法：jobs [选项] [jobspec…]

主要选项如下。

- -l：长输出用法，显示全部内容。
- -n：不输出信息。
- -p：只输出进程号。
- -r：只输出运行的进程。
- [jobspec]：后台任务号码。

应用实例如下。

先把两个进程放在系统后台运行，然后使用 jobs 命令查看后台任务的执行情况。

```
#du -a /etc > user.data &
[1] 233
# find / -name core -type f -ls > core.data &
[2] 234
#jobs –l
[1] + 237 Running du -a /etc > user.data
[2] - 238 Running find / -name core -type f -ls > core.data
```

说明

上面的当前任务是“du -a /etc > user.data”，因为后台任务号码是“[1]”。当第一个后台任务顺利执行完毕，第二个后台任务还在执行中时，当前任务便会自动变成后台任务号码“[2]”的后台任务，即当前任务是动态的。

4.11 kill：杀掉进程

作用：终止一个进程。

用法：kill [-s signal |-p] [-a]pid…

　　　kill -l [signal]

主要选项如下。

- -s：指定发送的信号。
- -p：模拟发送信号。
- -l：指定信号的名称列表。
- pid：要终止的进程的 ID 号。
- signal：信号。

说明

kill 可将指定的信息送至程序。预设的信息为 SIGTERM（15），可将指定程序终止。若仍无法终止该程序，可使用 SIGKILL（9）信息尝试强制删除程序。kill 命令的工作原理是：首先向 Linux 系统的内核发送一个系统操作信号和某个程序的进程标识号，然后系统内核就可以对进程标识号指定的进程进行操作。当需要中断一个前台进程的时候，通常使用 Ctrl+C 组合键；但是对于一个后台进程，就不是一个组合键所能解决的了，这时必须使用 kill 命令。

应用实例如下。

1）命令执行过程如果出错，用户可用“kill”命令来结束任务

对于在后台运行的进程，可以使用 kill 命令终止。

```
#du -a / | sort -rn > /tmp/du.sorted &
[1] 237
#kill 237
```

或者使用如下命令。

```
#du -a / | sort -rn > /tmp/du.sorted &
[1] 237
#kill  %1
```

（2）对于僵尸进程，可以用 kill-9 来强制终止退出

比如一个程序已经彻底死掉，如果 kill 不加信号强度没有办法退出，最好的办法就是加信号强度-9，后面要接杀父进程。示例代码如下：

```
# ps aux |grep gaim
beinan    5031  9.0  2.3 104996 17484 ?       S    13:23   0:01 gaim
root      5036  0.0  0.0   5160   724 pts/3   S+   13:24   0:00 grep gaim
```

kill 命令族成员简介如下。

终止一个进程或终止一个正在运行的程序，一般通过 kill、killall、pkill、xkill 等工具进行。比如一个程序已经死掉，但又不能退出，这时就应该考虑应用这些工具。killall 通过程序的名字直接杀死所有进程。pkill 和 killall 的应用方法差不多，也是直接杀死运行中的程序。如果想杀掉单个进程，可使用 kill。xkill 是在桌面用的杀死图形界面的程序。比如当 Firefox 出现崩溃不能退出时，单击就能杀死 Firefox。当 xkill 运行时光标处会出现一个人脑骨的图标，哪个图形程序崩溃了，单击该图标即可。如果想终止 xkill，则单击鼠标右键取消。

4.12 crontab：设置计时器

作用：安装、删除或者列出用于驱动 cron 后台进程的任务表，然后该配置由 cron 守护进程在设定的时间执行。

用法：crontab [-u user]文件

crontab [-u user] { -e | -l | -r }

主要选项如下。

- -e：执行文字编辑器来设定时程表，默认的文字编辑器是 vi。
- -r：删除目前的时程表。
- -l：列出目前的时程表。
- [-u user]：指定要设置的用户名称。

crontab 文件的用法如下：

```
minute hour day month dayofweek command
```

其中，minute——分钟，取 0～59 之间的任何整数。hour——小时，取 0～23 之间的任何整数。day——日期，取 1～31 之间的任何整数（如果指定了月份，则必须是该月份的有效日期）。month——月份，取 1～12 之间的任何整数（或使用月份的英文简写，如 Jan、Feb 等）。dayofweek——星期，取 0～7 之间的任何整数，这里的 0 或 7 代表星期日（或使用星期的英文简写，如 Sun、Mon 等）。command——要执行的命令（可以是 ls 之类的命令，也可以执行自

行编写的脚本命令)。

应用实例如下。

设置一个定时、定期的系统提示。

```
#crontab －e
```

此时系统会打开一个 vi 编辑器，如果输入 35 17 * * 5 Wall“明天休息，注意备份数据库”，然后存盘退出，这时在/var/spool/cron/目录下会产生一个 cao 文件，内容如下：

```
# DO NOT EDIT THIS FILE － edit the master and reinstall.
# (/tmp/crontab.2707 installed on Thu Jan  1 22:01:51 2004)
# (Cron version －－ $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
35 17 * * 5 wall "明天休息，注意备份数据库"
```

这样到每个星期五下班时间（17:35），系统就会弹出一个显示屏提醒用户注意备份数据库。显示结果如图 4-2 所示。

图 4-2　一个定时、定期的系统提示

4.13　ps：查看权限

作用：查看系统中进程的状态。

用法：ps [选项]

主要选项如下。

- -a：显示系统中所有进程的信息。
- -e：显示所有进程的信息。
- -f：显示进程的所有信息。
- -l：以长用法显示进程信息。
- -r：只显示正在运行的进程。
- -u：显示面向用户的用法（包括用户名、CPU 及内存使用情况等信息）。
- -x：显示所有非控制终端上的进程信息。
- -p：显示由进程 ID 指定的进程信息。
- -t：显示指定终端上的进程信息。

说明

要对进程进行监测和控制，首先要了解当前进程的情况，也就是需要查看当前进程。ps 命令是最基本、也是非常强大的进程查看命令。根据显示的信息可以确定哪个进程正在运行、哪个进程被挂起、进程已运行了多久、进程正在使用的资源、进程的相对优先级，以及进程的标识号（PID）。所有这些信息对用户都很有用，对于系统管理员来说更为重要。使用“ps aux”命令可以获得终端上所有用户的有关进程的所有信息。下面将结合图 4-3 讲解进程的基本信息。

```
[root@localhost ~]# ps aux
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  2068  616 ?        Ss   10:51   0:00 init [5]
root         2  0.0  0.0     0    0 ?        S<   10:51   0:00 [migration/0]
root         3  0.0  0.0     0    0 ?        SN   10:51   0:00 [ksoftirqd/0]
root         4  0.0  0.0     0    0 ?        S<   10:51   0:00 [watchdog/0]
root         5  0.0  0.0     0    0 ?        S<   10:51   0:00 [migration/1]
root         6  0.0  0.0     0    0 ?        SN   10:51   0:00 [ksoftirqd/1]
root         7  0.0  0.0     0    0 ?        S<   10:51   0:00 [watchdog/1]
root         8  0.0  0.0     0    0 ?        S<   10:51   0:00 [events/0]
root         9  0.0  0.0     0    0 ?        S<   10:51   0:00 [events/1]
root        10  0.0  0.0     0    0 ?        S<   10:51   0:00 [khelper]
root        11  0.0  0.0     0    0 ?        S<   10:51   0:00 [kthread]
root        15  0.0  0.0     0    0 ?        S<   10:51   0:00 [kblockd/0]
root        16  0.0  0.0     0    0 ?        S<   10:51   0:00 [kblockd/1]
root        17  0.0  0.0     0    0 ?        S<   10:51   0:00 [kacpid]
root       153  0.0  0.0     0    0 ?        S<   10:51   0:00 [cqueue/0]
```

图 4-3　ps aux 命令详解

在图 4-3 所示的第二行代码中，USER 表示启动进程用户。PID 表示进程标识号。%CPU 表示运行该进程占用 CPU 的时间与该进程总的运行时间的比例。%MEM 表示该进程占用内存和总内存的比例。VSZ 表示占用的虚拟内存大小，以 KB 为单位。RSS 表示进程占用的物理内存值，以 KB 为单位。TTY 表示该进程建立时所对应的终端，“?”表示该进程不占用终端。STAT 表示进程的运行状态，包括以下几种代码：D，不可中断的睡眠；R，就绪（在可运行队列中）；S，睡眠；T，被跟踪或停止；Z，终止（僵死）的进程；Z，不存在，但暂时无法消除；W，没有足够的内存分页可分配；<，高优先级的进程；N，低优先级的进程；L，有内存分页分配并锁在内存体内（实时系统或 I/O）。START 表示进程开始时间。TIME 表示执行的时间。COMMAND 表示对应的命令名。

应用实例如下。

（1）按内存占用情况对进程排序。

```
# ps auxw --sort=rss
```

（2）在进行系统维护时，如果 CPU 负载突然增加，而又不知道是哪一个进程造成的，可执行如下命令进行查看。

```
#ps auxw --sort=%cpu
```

4.14　pstree：显示进程状态树

作用：列出当前的进程，以及它们的树状结构。

用法：pstree [选项] [pid|user]

主要选项如下。

- -a：显示执行程序的命令与完整参数。
- -c：取消同名程序，合并显示。
- -h：对输出结果进行处理，高亮显示正在执行的程序。
- -l：以长用法显示。
- -n：以 PID 大小排序。
- -p：显示 PID。
- -u：显示 UID 信息。
- -G：使用 VT100 终端编码显示。
- -U：使用 UTF-8（Unicode）编码显示。

> **说明**
>
> 使用 ps 命令得到的数据精确，但数据庞大，这一点对于掌握系统整体概况来说是不容易的。pstree 命令正好可以弥补这个缺憾，它能将当前的执行程序以树状结构显示。pstree 命令支持指定特定程序（PID）或使用者（USER）作为显示的起始。

应用实例如下。

进程启动的时候可能会产生自己的一个子进程。运行 pstree 命令可以很容易地看到这些信息。以超级用户权限运行 pstree。

```
#init-+-apmd
     |-atd
     |-bdflush
     |-gconfd-2
     |-gdm-binary---gdm-binary-+-X
     |                         `-startkde-+-kwrapper
     |                                    `-ssh-agent
     |-gpm
|-httpd---8*[httpd]
…
```

pstree 命令对程序名称相同的会自动合并，“|-httpd---8*[httpd]”表示系统中有 8 个 httpd 进程产生的子进程。

4.15 top：显示进程

作用：显示系统当前的进程状况。

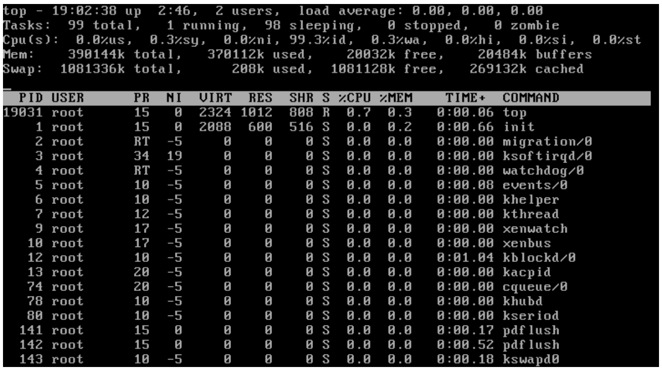
用法：top [选项]

主要选项如下。

- -b：使用批处理模式。
- -c：显示程序并显示程序的完整相关信息，如名称、路径等。
- -i：忽略闲置或已经冻结的程序。
- -d<delay>：以秒为单位，设定监控程序执行状况的时间间隔。
- -n<iterations>：设定监控信息的更新次数。
- -p<进程号>：指定进程。
- -s：安全模式。
- -u<somebody>：指定用户名。
- -v：显示版本信息。
- -h：显示帮助信息。

说明

top 命令和 ps 命令的基本作用是相同的，都是显示系统当前的进程状况。但是 top 命令是一个动态显示过程，即可以通过用户按键来不断刷新当前状态。下面结合图 4-4 来说明它给出的信息。



```
top - 19:02:38 up  2:46,  2 users,  load average: 0.00, 0.00, 0.00
Tasks:  99 total,   1 running,  98 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.3%sy,  0.0%ni, 99.3%id,  0.3%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    390144k total,   370112k used,    20032k free,    20484k buffers
Swap:  1081336k total,      208k used,  1081128k free,   269132k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
19031 root      15   0  2324 1012  808 R  0.7  0.3   0:00.06 top
    1 root      15   0  2088  600  516 S  0.0  0.2   0:00.66 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    4 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    5 root      10  -5     0    0    0 S  0.0  0.0   0:00.08 events/0
    6 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 khelper
    7 root      12  -5     0    0    0 S  0.0  0.0   0:00.00 kthread
    9 root      17  -5     0    0    0 S  0.0  0.0   0:00.00 xenwatch
   10 root      17  -5     0    0    0 S  0.0  0.0   0:00.00 xenbus
   12 root      10  -5     0    0    0 S  0.0  0.0   0:01.04 kblockd/0
   13 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
   74 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 cqueue/0
   78 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 khubd
   80 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 kseriod
  141 root      15   0     0    0    0 S  0.0  0.0   0:00.17 pdflush
  142 root      15   0     0    0    0 S  0.0  0.0   0:00.52 pdflush
  143 root      10  -5     0    0    0 S  0.0  0.0   0:00.18 kswapd0
```

图 4-4 top 命令详解

在图 4-4 中分成几行显示，其含义分别如下。

- 第一行表示的项目依次为当前时间、系统启动时间、当前系统登录用户数目、平均负载。
- 第二行显示的是所有启动的、目前运行的、挂起（Sleeping）的和无用（Zombie）的

进程。

- 第三行显示的是目前 CPU 的使用情况，包括系统占用的比例、用户使用比例、闲置（Idle）比例。
- 第四行显示物理内存的使用情况，包括总的可以使用的内存、已用内存、空闲内存、缓冲区占用的内存。
- 第五行显示交换分区的使用情况，包括总的、使用的、空闲的和用于高速缓存的交换分区。
- 第六行显示的项目最多，下面列出了详细解释。
  - Ø PID（Process ID）：进程标识号，是非零正整数。
  - Ø USER：进程所有者的用户名。
  - Ø PR：进程的优先级别。
  - Ø NI：进程的优先级别数值。
  - Ø VIRT：进程占用的虚拟内存值。
  - Ø RES：进程占用的物理内存值。
  - Ø SHR：进程占用的共享内存值。
  - Ø STAT：进程的状态，其中 S 表示休眠，R 表示正在运行，Z 表示僵死状态，N 表示该进程优先级别数值是负数。
  - Ø %CPU：该进程占用的 CPU 使用率。
  - Ø %MEM：该进程占用的物理内存和总内存的百分比。
  - Ø TIME：该进程启动后占用的总 CPU 时间。
  - Ø COMMAND：进程启动的启动命令名称。如果这一行显示不下，进程会有一个完整的命令行。

在 top 命令的使用过程中，还可以使用一些交互的命令来完成其他参数的功能。这些命令是通过以下快捷键启动的。

- <空格>：立刻刷新。
- P：根据 CPU 使用大小进行排序。
- T：根据时间、累计时间进行排序。
- q：退出 top 命令。
- m：切换显示内存信息。
- t：切换显示进程和 CPU 状态信息。
- c：切换显示命令名称和完整命令行。
- M：根据使用内存大小进行排序。
- W：将当前设置写入～/.toprc 文件中。这是写 top 配置文件的推荐方法。

可以看到，top 命令是一个功能十分强大的监控系统的工具，对于系统管理员而言尤其重要。但是，它的缺点是会消耗很多系统资源。

应用实例如下。

（1）显示程序及其完整的相关信息。

```
#top -c
```

（2）以秒为单位，设定监控程序执行状况的时间间隔。

例如，设定 top 命令的时间间隔为 3 秒，可在命令行提示符下输入如下命令：

```
#top -d 3
```

（3）设定监控信息的更新次数。

例如，设定系统任务信息更新 5 次后结束 top 命令，可在命令行提示符下输入如下命令：

```
#top -n 5 ↙
```

4.16 nice：改变优先权等级

作用：改变程序执行的优先权等级。

用法：nice [-n <优先等级>] [--help] [--version] [命令]

主要选项如下。

- -n<优先等级>或-<优先等级>或--adjustment=<优先等级>：设置欲执行的命令的优先权等级。等级的范围为–20～19，其中–20 为最高，19 为最低。
- --help：在线帮助。

> **说明**
>
> 应用程序优先权值的范围为–20～19，数字越小，优先权就越高。一般情况下，普通应用程序的优先权值（CPU 使用权值）都是 0，如果让常用程序拥有较高的优先权等级，自然，其启动和运行速度都会快些。需要注意的是，普通用户只能在 0～19 之间调整应用程序的优先权值，只有超级用户才有权调整更高的优先权值（–20～19）。

应用实例如下。

将 ls 命令的优先权等级加 1 并执行，将结果输入文件 1.txt。

```
# nice -n 1 ls >1.txt
```

4.17 renice：修改优先权等级

作用：renice 命令允许用户修改一个正在运行的进程的优先权等级。

用法：renicepriority[[-p]pids] [[-g]pgrps] [[-u]users]

主要选项如下。

- priority：优先权等级。

- -p pids：改变该程序的优先权等级，此参数为预设值。
- -g pgrps：使用程序群组名称，修改所有隶属于该程序群组的程序的优先权等级。
- -u user：指定用户名称，修改所有隶属于该用户的程序的优先权等级。

> **说明**
>
> renice 命令可重新调整正在执行的程序的优先权等级。默认情况下用程序识别码指定程序，调整其优先权等级，也可以指定程序群组或用户名称调整优先权等级，并修改所有隶属于该程序群组或用户的程序的优先权等级。等级范围为–20～19，只有超级用户可以改变其他用户程序的优先权和设置负数等级，普通用户只能对自己所有的进程使用 renice 命令。

应用实例如下。

将进程 PID 为 987 及 32 的进程与进程拥有者为 daemon 及 root 的优先权等级分别加 1。

```
#renice +1 987 -u daemon root -p 32
```

4.18　sleep：暂停进程

作用：使进程暂停执行一段时间。

用法：sleep number [选项]

主要选项如下。

- number：时间长度，后面可接 s、m、h 或 d。
- s：以秒为单位。
- m：以分钟为单位。
- h：以小时为单位。
- d：以天为单位。

> **说明**
>
> 如果没有指定时间，则以秒为单位。此命令大多用于 Shell 程序设计，使两条命令执行之间停顿指定的时间。

应用实例如下。

使进程先暂停 60 秒，然后查看哪个用户登录到系统中。

```
#sleep 60; who
```

4.19 nohup：用户退出系统之后继续工作

作用：确保执行程序能在用户退出系统之后继续工作。

用法：nohup 命令

> **说明**
>
> 一般退出 Linux 系统时，会把所有的程序全部结束掉，包括那些后台程序。但有时候，例如，用户正在下载一个很大的文件，但是因下班或有事需要先退出系统，希望退出系统后程序还能继续执行。这时，可以使用 nohup 命令使进程在用户退出后仍继续执行。同时这些进程都在后台执行（命令放到后台运行，nohup 必须与&操作同时使用），结果则会写到用户自己目录下的 nohup.out 文件中。

应用实例如下。

程序在后台自动执行。

```
nohup wget -c -t0 http://www.bsdr.com/ghs1.rar &
```

4.20 pgrep：查找匹配条件的进程

作用：查找当前运行的进程，并列出匹配给定条件的进程的 PID。所有的条件都必须匹配才会被列出。

用法：pgrep[选项] [程序名]

主要选项如下。

- -l：列出程序名和进程 ID。
- -o：进程起始的 ID。
- -n：进程终止的 ID。

应用实例如下。

用户 cao 查看 sshd 的进程列表。

```
[cao@localhost@cao]$pgrep –l sshd
829 sshd
```

4.21 fuser：用文件或者套接口表示进程

作用：用文件或者套接口表示进程。

用法：fuser [-a | -s | -c] [-4 | -6] [-n space] [-k [-i] [-signal]] [-muvf] name …

fuser -l

fuser -V

主要选项如下。

- -a：显示在命令行指定的所有文件，在默认情况下，至少被一个进程访问的文件才能显示出来。
- -c：同选项-m，用于同 POSIX 进行兼容。
- -f：忽略，用于同 POSIX 进行兼容。
- -i：结束进程前询问用户意见。
- -k：结束正在访问文件的所有进程。
- -l：列出所有已知的信号名字。
- -m：挂载文件系统。
- -n<space>：选择一个不同的名字空间，名字空间是指文件（默认为文件名）、udp 和 tcp。
- -s：不显示处理信息，选项-u 和-v 在此模式下将被忽略，选项-a 不能与该选项一起使用。
- -signal：结束进程时使用指定的信号而不是 SIGKILL；当不使用选项-k 时，该选项将被忽略。
- -u PID：显示用户名。
- -v：显示运行时的详细信息。
- -V：显示版本信息。

应用实例如下。

（1）列出所有已知的信号名字。

```
# fuser -l
HUP INT QUIT ILL TRAP ABRT IOT BUS FPE KILL USR1 SEGV USR2 PIPE ALRM TERM
STKFLT CHLD CONT STOP TSTP TTIN TTOU URG XCPU XFSZ VTALRM PROF WINCH IO PWR SYS
UNUSED
```

（2）显示进程。

例如，显示与/home/cjh/目录相关的所有进程。

```
# fuser -a /home/cjh
/home/cjh:          19169c 19197c
```

（3）结束正在访问文件的所有进程。

例如，结束正在访问目录/home/cjh/tmp/的所有进程。

```
#fuser -k /home/cjh/tmp/
/home/cjh/tmp/:     19169c
```

（4）显示用户名。

例如，显示所有访问目录/home/cjh/的进程，并显示进程的用户名。

```
# fuser -u /home/cjh/
```

```
/home/cjh/:         19245c(cjh)
```

（5）列出使用/etc/passwd 文件的本地进程的进程号。

```
#fuser /etc/passwd
```

（6）列出正在使用已从给定文件系统删除的文件的全部进程。

```
#fuser -d /usr
```

4.22 chkconfig：设置系统的各种服务

作用：检查、设置系统的各种服务。

用法：chkconfig [--add] [--del] [--list] [系统服务]

chkconfig [--level <等级代号>] [系统服务] [on/off/reset]

主要选项如下。

- --add：增加所指定的系统服务，让 chkconfig 指令得以管理它，并同时在系统启动的叙述文件内增加相关数据。
- --del：删除所指定的系统服务，不再由 chkconfig 指令管理，并同时在系统启动的叙述文件内删除相关数据。
- --level<等级代号>：指定读系统服务要在哪一个执行等级中开启或关闭。

说明

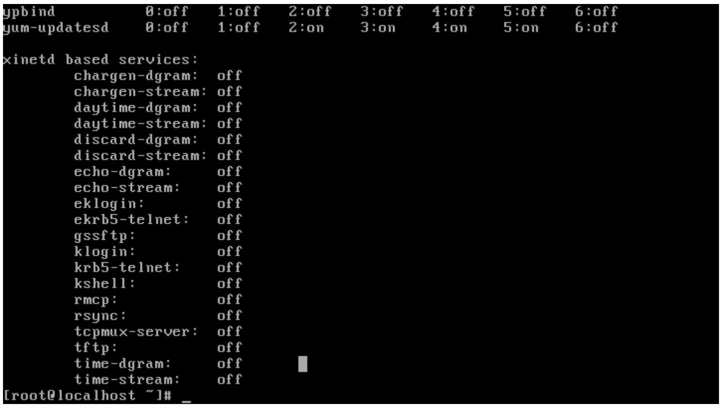
chkconfig 提供了一个简单的命令行工具用于维护/etc/rc[0-6].d 的路径层次，可以帮助系统管理员在这些路径中直接操作符号行。chkconfig 的执行是通过 chkconfig 命令激发的，此命令目前在 IRIX 操作系统中存在，甚至包括了维护/etc/rc[0-6].d 层次之外的设置信息。chkconfig 有 5 个不同的函数：为管理器添加新服务，从管理器中移出服务，列出当前启动的服务信息，改变服务启动信息，检查特殊服务的启动状态。这是 Red Hat 公司遵循 GPL 规则所开发的程序，它可查询操作系统在每个执行等级中会执行哪些系统服务，其中包括各类常驻服务。

应用实例如下。

（1）显示所有服务的状态。

```
#chkconfig –list
```

结果如图 4-5 所示。



```
ypbind          0:off   1:off   2:off   3:off   4:off   5:off   6:off
yum-updatesd    0:off   1:off   2:on    3:on    4:on    5:on    6:off

xinetd based services:
        chargen-dgram:  off
        chargen-stream: off
        daytime-dgram:  off
        daytime-stream: off
        discard-dgram:  off
        discard-stream: off
        echo-dgram:     off
        echo-stream:    off
        eklogin:        off
        ekrb5-telnet:   off
        gssftp:         off
        klogin:         off
        krb5-telnet:    off
        kshell:         off
        rmcp:           off
        rsync:          off
        tcpmux-server:  off
        tftp:           off
        time-dgram:     off
        time-stream:    off
[root@localhost ~]# _
```

图 4-5　显示所有服务的状态

结果会看到一个系统服务列表，以及它们在运行级别 0～6 中已被启动（on）或停止（off）。

（2）chkconfig 管理服务技巧。

从图 4-5 所示的列表末端会看到由 xinetd 管理的服务部分。如果使用 chkconfig--list 来查询由 xinetd 管理的服务，会看到 xinetd 服务是被启用了（on）还是被关闭了（off）。例如，命令 chkconfig--list finger 返回了下列输出：

```
finger          on
```

如上所示，finger 作为 xinetd 服务被启用。如果 xinetd 在运行，finger 就会被启用。

如果使用 chkconfig --list 来查询/etc/rc.d 中的服务，会看到服务在每个运行级别中的设置。例如，命令 chkconfig --list anacron 返回了下列输出：

```
Anacron         0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

chkconfig 还能用来设置某一服务在某一指定的运行级别内是被启动还是被停运。例如，要在运行级别 3、4、5 中停运 nscd 服务，可使用下面的命令：

```
Chkconfig --level 345 nscd off
```

说明

由 xinetd 管理的服务会立即被 chkconfig 影响。例如，如果 xinetd 在运行，finger 被禁用，那么执行了 chkconfig finger on 命令后，finger 可以随 xinetd 的改变而自动启用。对其他服务的改变在使用 chkconfig 之后不会立即生效，必须使用 service daemon stop 命令来停止个别服务。在前面的例子中，可以把 daemon 换成想停止的服务名称（如 httpd）、把 stop 换成 start 或 restart 来启动或重新启动该服务。

4.23 strace：跟踪一个进程的系统调用或信号产生的情况

作用：strace 命令是一个强大的工具，它能够显示所有由用户空间程序发出的系统调用。strace 显示这些调用的参数并返回符号形式的值。strace 从内核接收信息，而且不需要以任何特殊的方式来构建内核。

用法：strace [选项]

主要选项如下。

- -f：除了跟踪当前进程外，还跟踪其子进程。
- -o file：将输出信息写到文件 file 中，而不是显示到标准错误输出（stderr）。
- -p pid：绑定到一个由 PID 对应的正在运行的进程，此参数常用来调试后台进程。

应用实例如下。

（1）跟踪 vim 及其子进程的运行，将输出信息写到文件 vim.strace。

```
# strace -f -o vim.strace vim
```

（2）同时跟踪多个进程。

```
#strace -o a.strace -f -F -p 4579 -p 4580 -p 4581
```

4.24 ltrace：跟踪进程调用库函数的情况

作用：跟踪进程调用库函数的情况。

用法：ltrace [选项]

主要选项如下。

- -f：除了跟踪当前进程外，还跟踪其子进程。
- -o file：将输出信息写到文件 file 中，而不是显示到标准错误输出（stderr）。
- -p pid：绑定到一个由 PID 对应的正在运行的进程，此参数常用来调试后台进程。
- -d：打印调试信息。

应用实例如下。

跟踪一个 PID 为 234 的正在运行的进程。

```
#ltrace -p 234
```

4.25 vmstat：报告虚拟内存统计信息

作用：报告关于内核线程、虚拟内存、磁盘、陷阱和 CPU 活动的统计信息。由 vmstat 命令生成的报告可以用于平衡系统负载活动。系统范围内的这些统计信息（所有的处理器中）都

计算出以百分比表示的平均值，或者计算其总和。

用法：vmstat [-V] [-n] [延时[次数]]

主要选项如下。

- -a：显示活跃和非活跃内存。
- -f：显示从系统启动至今的 fork 数量。
- -m：显示 slabinfo。
- -n：只在开始时显示一次各字段名称。
- -s：显示内存相关统计信息及多种系统活动数量。
- delay：刷新时间间隔。如果不指定，只显示一条结果。
- count：刷新次数。如果不指定刷新次数，但指定了刷新时间间隔，这时刷新次数为无穷。
- -d：显示磁盘相关统计信息。
- -p：显示指定磁盘分区统计信息
- -S：使用指定单位显示。参数有 k、K、m、M，分别代表 1000、1024、1000000、1048576 字节（byte）。默认单位为 K（1024 bytes）。
- -V：显示 vmstat 版本信息。

应用实例如下。

使用 vmstat 5 5（表示在 5 秒内进行 5 次采样）命令测试。

```
#vmstat 5 5
   procs                    memory      swap        io     system      cpu
 r  b  w   swpd   free   buff  cache   si   so    bi    bo   in     cs us sy id
 0  0  0      0  32188   6412  47252    0    0   200    47  121     89  8  6 85
 0  0  0      0  32188   6420  47252    0    0     0     6  101     20  0  0 100
 0  0  0      0  32188   6420  47252    0    0     0     0  101     19  0  0 100
 0  0  0      0  32188   6420  47252    0    0     0     0  101     20  0  0 100
 0  0  0      0  32188   6420  47252    0    0     0     0  101     22  0  0 100
```

结果将得到一个数据汇总，它可以反映真正的系统情况。vmstat 命令输出分成 6 个部分。

1）procs（进程）

- r：在运行队列中等待的进程数。
- b：在等待 I/O 的进程数。
- w：可以进入运行队列但被替换的进程。

2）memory（内存）

- swpd：现时可用的交换内存，单位为 KB。
- free：空闲的内存，单位为 KB。
- buff：缓冲区中的内存，单位为 KB。
- cache：被用来作为高速缓存的内存，单位为 KB。

3）交换页面（swap）

- si：从磁盘交换到内存的交换页数量，单位为 KB/s。

– so：从内存交换到磁盘的交换页数量，单位为 KB/s。

4）I/O（块设备）

– bi：发送到块设备的块数，单位为块/s。

– bo：从块设备接收到的块数，单位为块/s。

5）system（系统）

– in：每秒的中断数，包括时钟中断。

– cs：每秒的环境（上下文）切换次数。

6）CPU（中央处理器）

– cs：用户进程使用的时间，以百分比表示。

– sy：系统进程使用的时间，以百分比表示。

– id：中央处理器的空闲时间，以百分比表示。

如果 r 经常大于 4，且 id 经常小于 40，则表示中央处理器的负荷很重；如果 bi 和 bo 长期不等于 0，则表示物理内存容量太小。

4.26 mpstat：监测 CPU（包括多 CPU）性能

作用：mpstat 命令专门监测 CPU（包括多 CPU）性能。

用法：mpstat[选项]

主要选项如下。

– -P {|ALL}：表示监控哪个 CPU，CPU 在[0,CPU 个数–1]中取值。

– internal：相邻两次采样的间隔时间。

– count：采样的次数，count 只能和 delay 一起使用。

应用实例如下。

```
# mpstat
Linux 2.6.18-164.el5xen (localhost.localdomain)       2010年03月26日
16时28分28秒 CPU %user %nice %sys %iowait %irq %soft %steal %idle intr/s
16时28分28秒 all  0.11  0.02  0.16  0.35  0.00 0.03   0.03 99.31  53.81
```

mpstat 输出说明如下。

– CPU：处理器 ID。

– %user：用户模式（应用程序）占用 CPU 的百分比。

– %nice：用户模式（使用 nice 命令改变既定优先级权限的应用程序）占用 CPU 的百分比。

– %system：核心占用 CPU 的百分比。

– %idle：处理器的闲置时间百分比。

以上 4 项总和为 100%。

– intr/s：处理器每秒处理的中断数量。

4.27 iostat：监测 I/O 性能

作用：iostat 是 I/O statistics（输入/输出统计）的缩写，iostat 工具将对系统的磁盘操作活动进行监视。它的特点是不仅汇报磁盘活动统计情况，同时也汇报 CPU 使用情况。同 vmstat 一样，iostat 也有一个弱点，就是它不能对某个进程进行深入分析，仅对系统的整体情况进行分析。

用法：iostat[-c|-d] [-k] [-t] [-V] [-x[device]] [interval[count]]

主要选项如下。

- -c：显示 CPU 的使用情况。
- -d：显示磁盘的使用情况。
- -k：按千字节每秒显示数据。
- -t：打印汇报的时间。
- -v：打印出版本信息和用法。
- -x device：指定要统计的设备名称，默认为所有的设备。
- interval：指每次统计间隔的时间，count 指按照这个时间间隔统计的次数。

说明

目前 Linux 操作系统的主要性能瓶颈在于 I/O 环节，这主要是由于 I/O 系统性能的提高远远低于 CPU 和内存性能的提高。根据摩尔定律，CPU 性能和内存容量每 18 个月就可翻一番。但是作为外存的主要设备——硬磁盘，由于机械运动的本质特征导致性能的提高非常有限，每年约 7%。这样可由摩尔定律得出，如今 I/O 的性能在系统整体性能中占有举足轻重的地位。目前优化磁盘性能的各种技术主要侧重于如何提高数据传输的带宽，很少有侧重于降低延迟的。

应用实例如下。

用 iostat -x /dev/sda1 查看磁盘 I/O 的详细情况。

```
# iostat –x /dev/sda1
Linux 2.6.18-164.el5xen (localhost.localdomain)       2010年03月26日
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          0.11    0.02    0.18    0.35     0.03    99.31
Device:         tps   Blk_read/s    Blk_wrtn/s    Blk_read   Blk_wrtn
sda1           0.02         0.08          0.00       2014          4
```

详细说明：第二行是系统信息和监测时间，第三行和第四行显示 CPU 使用情况（具体内容和 mpstat 命令相同）。这里主要关注后面 I/O 输出的信息，如表 4-1 所示。

表 4-1　iostat 输出

标　　识	说　　明
Device	监测设备名称
rrqm/s	每秒需要读取需求的数量
wrqm/s	每秒需要写入需求的数量
r/s	每秒实际读取需求的数量
w/s	每秒实际写入需求的数量
rsec/s	每秒读取区段的数量
wsec/s	每秒写入区段的数量
rkB/s	每秒实际读取的大小，单位为 KB
wkB/s	每秒实际写入的大小，单位为 KB
avgrq-sz	需求的平均大小区段
avgqu-sz	需求的平均队列长度
await	等待 I/O 的平均时间（milliseconds）
svctm	I/O 需求完成的平均时间
%util	被 I/O 需求消耗的 CPU 百分比

4.28 sar：系统活动情况报告

作用：sar 是一个开放源代码工具，它由 Sebastien Godard 维护。这个工具也包含于大部分 Linux 发行版本中，可用于当前的 2.4 和 2.6 版内核，在 Red Hat Linux 中是 Sysstat。Sysstat 包括 I/O 与 CPU 统计资料的工具 iostat、mpstat（用于多处理器性能监测）和 sar。

用法：sar [选项]

主要选项如下。

- -A：汇总所有的报告。
- -a：报告文件读/写的使用情况。
- -B：报告附加的缓存的使用情况。
- -b：报告缓存的使用情况。
- -c：报告系统调用的使用情况。
- -d：报告磁盘的使用情况。
- -g：报告串口的使用情况。
- -h：报告关于 buffer 使用的统计数据。
- -m：报告 IPC 消息队列和信号量的使用情况。
- -n：报告命名 Cache 的使用情况。
- -p：报告调页活动的使用情况。
- -q：报告运行队列和交换队列的平均长度。

- -R：报告进程的活动情况。
- -r：报告没有使用的内存页面和硬盘块。
- -u：报告 CPU 的利用率。
- -v：报告进程、I 节点、文件和锁表状态。
- -w：报告系统交换活动状况。
- -y：报告 TTY 设备活动状况。

应用实例如下。

（1）查看内存和交换空间的使用率，使用 sar -r。

```
#  sar -r
Linux 2.4.20-8 (www.cao.com)   20050503
12:00:01 AM kbmemfree kbmemused  %memused kbmemshrd kbbuffers  kbcached
12:10:00 AM    240468   1048252    81.34        0    133724    485772
12:20:00 AM    240508   1048212    81.34        0    134172    485600
…
08:40:00 PM    934132    354588    27.51        0     26080    185364
Average:      324346    964374    74.83        0     96072    467559
```

kbmemfree 与 kbmemused 字段分别显示内存的未使用与已使用空间，后面跟着的是已使用空间的百分比（%memused 字段）。kbbuffers 与 kbcached 字段分别显示缓冲区与系统全域的数据存取量，单位为 KB。

（2）观察系统部件 10 分钟，并对数据进行排序。

```
#sar  -o temp 60 10
```

4.29 pidof：查找正在运行进程的进程 ID（PID）

作用：pidof 是 Linux 下用来查找正在运行进程的进程 ID (PID)。大部分使用 pgrep,ps 来代替。

用法：pidof [选项]

主要选项如下。

- -s：返回多进程，返回一个 PID。
- -c：只返回进程 ID。
- -x：表示同时返回运行给定程序的 Shell 的 PID。
- -o<进程号>：-o 表示告诉 piod 表示忽略后面给定的 PID，可以使用多个-o。可以用%PPID 表示忽略 pidof 程序的父进程的 PID，也就是调用 pidof 的 Shell 或者脚本的 pi。

说明

sar 是 System Activity Reporter（系统活动情况报告）的缩写。顾名思义，sar 工具将对系统当前的状态进行取样，然后通过计算数据和比例来表达系统的当前运行状态。它的特点是可以连续对系统取样，获得大量的取样数据。取样数据和分析的结果都可以存入文件，使用它时消耗的系统资源很小。sar 是通过读/proc 这个内存文件系统进行采样来得到数据的。sar 从/var/log/sa/sadd 读取记录文件的资料。/usr/lib/sa/sadc 负责维护系统活动资料和建立这些记录文件。sar 实际包括两个 Shell 程序：/usr/lib/sa/sa1 和/usr/lib/sa/sa2。/etc/cron.d/systat 是 crontab 的系统文件，每 10 分钟执行 sa1 程序一次，然后将它的输出加到记录文件后面。sa1 以二进制格式存储 sar 所读取的资料。/usr/lib/sa/sa2 将每天的数据写入文件/var/log/sa/sa*dd*（*dd* 表示每月的日期）。sar 是目前 Linux 上最为全面的系统性能分析工具之一，可以从 14 个大方面对系统的活动进行报告，包括文件的读/写情况、系统调用的使用情况、串口、CPU 效率、内存使用情况、进程活动及 IPC 有关的活动等，使用也较为复杂。

应用实例如下。

查看以下进程。

```
#pidof  -s nginx
 13312
```

4.30 ntsysv：设置系统服务

作用：ntsysv 是 Red Hat 公司遵循 GPL 规则所开发的程序，它具有交互式操作界面，可以轻易地利用方向键和空格键等，开启、关闭操作系统在每个执行等级中所要执行的系统服务。

用法：ntsysv [--back] [--level <levels>]

主要选项如下。

- --back：在交互式界面中显示上一步按钮。
- --level<levels>：在指定的执行等级中选择要打开或关闭的系统服务；等级为 0～6，可一次指定多个等级，如“123”表示同时设定等级为 1、2 和 3 的服务。

levels 执行的状态如下。

- 0：关机。
- 1：单人用户模式的文字界面。
- 2：多人用户的文字界面，但不具有 NFS 功能。
- 3：多人用户的文字界面，并且具有全部的网络功能。
- 4：某些发行版本使用此参数可进入 X Window。

– 5：某些发行版本使用此参数可进入 X Window。
– 6：重新启动。

应用实例如下。

（1）配置系统的各种服务运行等级

```
#ntsysv
```

如图 4-6 所示，可以按 F1 键寻求操作帮助。

（2）配置运行级别

编辑运行级别 0、1 和 6。

```
#ntsysv --levels 016
```

（3）在交互式界面中显示 Back 按钮

例如，配置系统的各种服务运行等级，并且在交互式界面中显示 Back 按钮，可在命令行提示符下输入：

```
#ntsysv --back
```

如图 4-7 所示，原来界面中的 Cancel 按钮已被 Back 按钮取代。

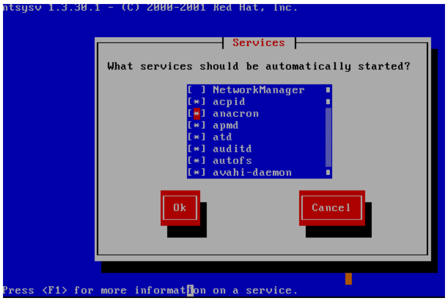


图 4-6　配置系统的各种服务运行等级

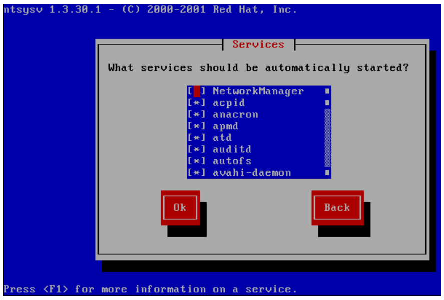


图 4-7　在交互式界面中显示 Back 按钮

05 Linux 网络管理命令

5.1 arp：管理系统中的 ARP 高速缓存

作用：管理系统中的 ARP 高速缓存。

用法：arp [选项]

主要选项如下。

- -a：使用 BSD 用法显示主机名称。
- -e：使用 Linux 用法显示主机名称。
- -d：删除 IP 地址和 MAC 地址对应的缓存记录。
- -i,--device：显示指定设备的缓存（比如 eth0）。
- -f,--file：从文件中读取数据。
- -s,--set：添加一个 IP 地址和 MAC 地址对应的记录。
- -n：不显示解析信息。
- -H<地址类型>：指定 arp 指令使用的地址类型。
- -D：使用指定接口的硬件地址。
- -v：显示详细的 ARP 缓冲区条目，包括缓冲区条目的统计信息。

说明

ARP，全称 Address Resolution Protocol，中文名为地址解析协议，它工作在数据链路层，在本层和硬件接口联系，同时对上层提供服务。IP 数据包常通过以太网发送，以太网设备并不识别 32 位 IP 地址，它们以 48 位以太网地址传输以太网数据包。因此，必须把 IP 目的地址转换成以太网目的地址。在以太网中，一个主机要和另一个主机进行直接通信，必须知道目标主机的 MAC 地址。但这个目标 MAC 地址是如何获得的呢？它就是通过地址解析协议获得的。ARP 用于将网络中的 IP 地址解析为硬件地址（MAC 地址），以保证通信的顺利进行。ARP 命令可以列出、修改、增加与删除内核中的 ARP 高速缓存数据。系统首先会参考/etc/hosts 文件，若查不到对应的名称，则向 DNS 服务器查询。若仍查不到，则会出现问号“?”。

应用实例如下。

（1）列出 ARP 缓存中所有的数据。

```
# arp –a
cjh  (192.168.40.11) at 00:50:56:C0:00:01 [ether] on eth0
```

（2）强迫以 IP 来显示与网卡对应的物理地址。

```
# arp -n
Address              HWtype  HWaddress           Flags Mask       Iface
192.168.118.254      ether   00:50:56:E7:89:7C      C              eth0
192.168.118.1        ether   00:50:56:C0:00:01      C              eth0
```

（3）修改网卡 MAC 地址。

```
# arp -s 192.168.40.1 00:01:80:0E:64:C5
# arp
Address              HWtype  HWaddress          Flags Mask        Iface
192.168.40.1         ether   00:01:80:0E:64:C5     CM              eth0
192.168.40.11        ether   00:50:56:C0:00:01     C               eth0
```

（4）通过 MAC 地址反查 IP 地址，假设 MAC 地址是 00:50:56:E7:89:7C。

```
# arp -a|grep 00:50:56:E7:89:7C
cjh (192.168.118.254) at 00:50:56:E7:89:7C [ether] on eth0
```

5.2 arpwatch：监听 ARP 记录

作用：监听网络上的 ARP 记录。

用法：arpwatch[-dN] [-fdatafile] [-iinterface] [-nnet[/width]] [-rfile] [-uusername] [-eusername]

主要选项如下。

- -d：启动排错模式。
- -f datafile：设置存储 ARP 记录的文件，预设为/var/arpwatch/arp.dat。
- -i interface：指定监听 ARP 的接口，预设的接口为 eth0。
- -r file：从指定的文件中读取 ARP 记录，而不是从网络上监听。
- -u username：指定可以使用该命令的用户，默认情况下是所有用户。
- -e username：将监听到的变化通过 E-mail 报告给用户。默认情况下是 root 用户。
- -s：指定用户名作为返回地址，而不是默认的用户 root。

说明

ARP 是用来解析 IP 与网络装置硬件地址的协议。arpwatch 命令可监听局域网络中的 ARP 数据包并记录，同时将监听到的变化通过 E-mail 报告。

应用实例如下。

利用排错模式执行 arpwatch，若在网络上检测到新的 ARP 记录，则将结果输出到屏幕。

```
# arpwatch -d
From: root (Arpwatch)
To: root
Subject: new station (cjh)
            hostname: cjh
          ip address: 192.168.40.135
    ethernet address: 0:c:29:50:2f:94
     ethernet vendor: VMware, Inc.
           timestamp: Wednesday, August 6, 2008 11:51:35 +0800
From: root (Arpwatch)
To: root
```

5.3 arping：发送 ARP 请求到一个相邻主机

作用：arping 是用于发送 ARP 请求到一个相邻主机的工具，arping 使用 ARP 数据包，通过 ping 命令检查设备上的硬件地址。

用法：arping [-fqbDUAV] [-c count] [-w timeout] [-I device] [-s source] destination

主要选项如下。

- -c count：表示发送指定数量的 ARP 请求数据包后就停止。如果指定了 deadline 选项，则 arping 会等待相同数量的 ARP 响应包，直到超时为止。
- -b：用于发送以太网广播帧（FFFFFFFFFFFF）。arping 一开始使用广播地址，在收到响应后就使用 unicast 地址。
- -w timeout：设定一个超时时间，单位是秒。如果到了指定时间，arping 还没有完全收到响应，则退出。
- -q：quiet output，不显示任何信息。
- -f：表示在收到第一个响应包后就退出。
- -s source：设定 arping 发送的 ARP 数据包中的 SPA 字段的值。如果为空，是 Unsolicited ARP 模式（Gratutious ARP），则设置为目标地址；否则从路由表得出。如果是 DAD 模

式（冲突地址探测），则设置为 0.0.0.0。

- -I<网卡接口>：使用指定的以太网设备，默认情况下使用 eth0，也就是设置 ping 使用的网络接口。
- -U：主动的 ARP 模式，更新邻居。
- -A：ARP 回复模式，更新邻居。
- destination：设置目标地址。

应用实例如下。

向 IP 地址为 192.168.118.129 的主机发送 3 次 ARP 请求数据包。

```
# arping -c 3 -w 3  -I eth0 192.168.118.129
ARPING 192.168.118.129 from 192.168.118.132 eth0
Sent 4 probes (4 broadcast(s))
Received 0 response(s)
```

5.4 arpd：收集免费 arp 信息的一个守护进程

作用：arpd 命令用来收集免费 arp 信息的一个守护进程，它将收集到的信息保存在磁盘上，或者在需要时提供给内核用户用于避免多余广播。

用法：arpd[-lkh?] [-aN] [-bdbase] [-Bnumber] [-ffile] [-ntime] [-pinterval] [-Rrate] [interfaces]

主要选项如下。

- -l：将 arp 数据库输出到标准输出设备显示并退出。
- -f：指定读取和加载 arpd 数据库的文本文件，文件的格式与“-l”输出信息类似。
- -b：指定 arpd 数据库文件，默认的位置为“/var/lib/arpd.db”。
- -a：指定目标被认为死掉前查询的次数。
- -k：禁止通过内核发送广播查询。
- -n：设定缓冲失效时间。

应用实例如下。

启动 arpd 进程。

```
arpd -b /var/tmp/arpd.db
```

运行一段时间后，查看结果。

```
arpd -l -b /var/tmp/arpd.db
```

5.5 finger：查找并显示用户信息

作用：finger 命令用来查询一台主机上的登录账号的信息，通常会显示用户名、主目录、

停滞时间、登录时间、登录 Shell 等信息。

用法：finger [选项] [使用者] [用户@主机]

主要选项如下。

- -s：显示用户注册名、实际姓名、终端名称、写状态、停滞时间、登录时间等信息。
- -l：除了用-s 选项显示的信息外，还显示用户主目录、登录 Shell、邮件状态等信息，以及用户主目录下的.plan、project 和.forward 文件的内容。
- -p：除了不显示.plan 文件和.project 文件以外，与-l 选项相同。
- -m：排除查找用户的真实姓名。

> **说明**
>
> 如果要查询远程主机上的用户信息，需要在用户名后面接“@主机名”，采用“用户名@主机名”的用法，不过要查询的网络主机需要有运行 finger 守护进程的支持。

应用实例如下。

在计算机上查找并显示用户信息。

```
# finger -s
Login     Name      Tty      Idle  Login Time   Office     Office Phone
root      root      tty2        7  Mar 25 16:24
root      root      pts/1    2:03  Mar 25 23:43 (192.168.118.1)
root      root      pts/2    1:19  Mar 26 00:46 (192.168.118.1)
root      root      pts/3    2:10  Mar 25 20:33 (192.168.118.1)
root      root      pts/4          Mar 26 01:17 (192.168.118.1)
```

5.6 ifconfig：设置网络接口

作用：设置网络设备的状态，或显示目前的设置。

用法：ifconfig [网络设备]选项

主要选项如下。

- add <address>：设置网络设备的 IP 地址。
- del <address>：删除网络设备的 IP 地址。
- hw <HW> <address> metric <NN> mtu <NN>：设置网络设备的类型与硬件地址。
- io_addr <NN>：设置网络设备的 I/O 地址。
- irq <NN>：设置网络设备的 IRQ。
- media <type>：设置网络设备的媒介类型。
- mem_start <NN>：设置网络设备在主内存中所占用的起始地址。
- metric <NN>：指定在计算数据包的传送次数时所要加上的数目。

- Mtu <NN>：设置网络设备的 MTU。
- Netmask <address>：设置网络设备的子网掩码。
- Tunnel <address>：建立 IPv4 与 IPv6 之间的隧道通信地址。
- up：启动指定的网络设备。
- down：关闭指定的网络设备。
- [网络设备]–promisc：关闭或启动指定网络设备的 promiscuous 模式。
- -broadcast <address>：将要送往指定地址的数据包当成广播数据包来处理。
- -pointopoint <address>：与指定地址的网络设备建立直连，此模式具有保密功能。
- -promisc：设置是否支持网卡的 promiscuous 模式；如果选择此参数，网卡将接收网络中发给它所有的数据包。
- -allmulti：设置是否支持多播模式；如果选择此参数，网卡将接收网络中所有的多播数据包。
- -a：显示全部接口信息。
- -s：显示摘要信息（类似于 netstat -i）。
- dstaddr：设定一个远端地址，建立点对点通信。
- -broadcast<地址>：为指定网卡设置广播协议。
- multicast：为网卡设置组播标志。
- txqueuelen<长度>：为网卡设置传输队列的长度。

说明

ifconfig 命令是比较重要的 Linux 网络管理命令，其最主要的用途是修改网卡的 IP 地址。

应用实例如下。

（1）修改网卡 IP 地址。

```
#ifconfig eth0 192.168.0.2 netmask 255.255.255.0
```

（2）为一个网卡设定两个 IP 地址。

```
#ifconfig eth0：1 192.168.0.2 netmask 255.255.255.0
#ifconfig eth0：2 192.168.0.5 netmask 255.255.255.0
```

（3）为网卡添加一个 IPv6 地址。

```
#modprobe ipv6“加载 IPv6 模块”
#ifconfig eth0 inet6 add 3ffe:ffff:0:f101::1/64
```

（4）使用 ifconfig 命令修改网卡 MAC 地址。

```
#ifconfig eth0 down
#ifconfig eth0 ether 00:06:11:4e:ff
#ifconfig eth0 up
```

5.7 iwconfig：设置无线网卡

作用：iwconfig 命令是 LWE 最主要的工具，可以对无线网卡的大部分参数进行配置。

用法：iwconfig interface [essid {NN|on|off}]
    [nwid {NN|on|off}]
    [mode {managed|ad-hoc|...}
    [freq N.NNNN[k|M|G]]
    [channel N]
    [ap {N|off|auto}]
    [sens N]
    [nick N]
    [rate {N|auto|fixed}]
    [rts {N|auto|fixed|off}]
    [frag {N|auto|fixed|off}]
    [enc {NNNN-NNNN|off}]
    [power {period N|timeout N}]
    [retry {limit N|lifetime N}]
    [txpower N {mW|dBm}]
    [commit]
    key/enc[ryption]

主要选项如下。

- essid：设置无线网卡的 ESSID（Extension Service Set ID）。通过 ESSID 来区分不同的无线网络，正常情况下只有 ESSID 相同的无线站点才可以互相通信，除非想监听无线网络。其后的参数为双引号括起的 ESSID 字符串，或者是 any/on/off。如果 ESSID 字符串中包含 any/no/off，则需要在前面加“--”。示例如下：

```
#iwconfig eth0 essid any                //允许任何 ESSID，也就是混杂模式
#iwconfig eth0 essid "My Network"       //设置 ESSID 为“My Network”
#iwconfig eth0 essid -- "ANY"           //设置 ESSID 为“ANY”
```

- nwid：Network ID，只用于 pre-802.11 的无线网卡，802.11 网卡利用 ESSID 和 AP 的 MAC 地址来替换 nwid，现在基本上不用设置。示例如下：

```
#iwconfig eth0 nwid AB34
#iwconfig eth0 nwid off
```

- mode：设置无线网卡的工作模式，可以是如下模式。
  - Ø Ad-hoc：不带 AP 的点对点无线网络。
  - Ø Managed：通过多个 AP 组成的网络，无线设备可以在这个网络中漫游。
  - Ø Master：设置该无线网卡为一个 AP。

- Repeater：设置为无线网络中继设备，可以转发网络包。
- Secondary：设置为备份的 AP/Repeater。
- Monitor：监听模式。
- Auto：由无线网卡自动选择工作模式。

示例如下：

```
#iwconfig eth0 mode Managed
#iwconfig eth0 mode Ad-Hoc
```

- freq/channel：设置无线网卡的工作频率或者频道，小于 1000 的参数被认为是频道，大于 1 000 的参数被认为是频率。频率的单位为 Hz，可以在数字后面附带 k、M、G 来改变数量级，如 2.4G。频道从 1 开始。使用 iwlist 工具可以查看无线网卡支持的频率和频道。参数 off/auto 指示无线网络自动挑选频率。注意，如果是 Managed 模式，AP 会指示无线网卡的工作频率，因此设置的参数会被忽略。Ad-hoc 模式下只使用该设定的频率初始化无线网络，如果加入已经存在的 Ad-hoc 网络，则会忽略设置的频率参数。示例如下：

```
#iwconfig eth0 freq 2422000000
#iwconfig eth0 freq 2.422G
#iwconfig eth0 channel 3
#iwconfig eth0 channel auto
```

- ap：连接到指定的 AP 或者无线网络，后面的参数可以是 AP 的 MAC 地址，也可以是 iwlist 扫描出来的标识符。如果是 Ad-hoc，则连接到一个已经存在的 Ad-hoc 网络。使用 off 参数让无线网卡不改变当前已连接的 AP 进入自动模式。使用 any/auto 参数，无线网卡将自动选择最好的 AP。

> **注意**
> 如果无线信号低到一定程度，则无线网络会进入自动选择 AP 模式。

示例如下：

```
#iwconfig eth0 ap 00:60:1D:01:23:45
#iwconfig eth0 ap any
#iwconfig eth0 ap off
```

- sens：设置接收灵敏度的下限，在该下限之下，无线网卡认为该无线网络信号太差，不同的网卡会采取不同的措施，一些现代的无线网卡会自动选择新的 AP。正值为 raw data，直接传给无线网卡驱动处理，一般认为是百分比；负值表示 dBm 值。示例如下：

```
#iwconfig eth0 sens -80
#iwconfig eth0 sens 2
```

- nick：nickname，一些网卡需要设置该参数，但是 802.11 协议栈、MAC 都没有用到该参数，一般也不用设置。示例如下：

```
#iwconfig eth0 nickname "My Linux Node"
```

– rate/bit：如果无线网卡支持多速率，则可以通过该命令设置工作的速率。小于 1000 的参数由具体的无线网卡驱动定义，一般是传输速率的索引值；大于 1000 的为速率，单位为 bit/s，可以在数字后面附带 k、M、G 来指定数量级。auto 参数让无线网卡自动选择速率；fixed 参数让无线网卡不使用自动速率模式。示例如下：

```
#iwconfig eth0 rate 11M
#iwconfig eth0 rate auto
#iwconfig eth0 rate 5.5M auto    //自动选择 5.5Mbit/s 以下的速率
```

– rts：指定 RTS/CTS 握手方式。使用 RTS/CTS 握手会增加额外开销，但当无线网络中有隐藏无线节点或者有很多无线节点时可以提高性能。后面的参数指定一个使用该机制的最小包的大小；如果该值等于最大包大小，则禁止使用该机制。可以使用 auto/off/fixed 参数。示例如下：

```
#iwconfig eth0 rts 250
#iwconfig eth0 rts off
```

– frag：设置发送数据包的分片大小。设置分片会增加额外开销，但在噪声环境下可以提高数据包的到达率。一般情况下该参数小于最大包大小，有些支持 Burst 模式的无线网卡可以设置大于最大包大小的值来允许 Burst 模式。还可以使用 auto/fixed/off 参数。示例如下：

```
#iwconfig eth0 frag 512
#iwconfig eth0 frag off
```

– txpower：如果无线网卡支持多发射功率设定，则使用该参数设定发射，单位为 dBm。如果指定为 *W*（毫瓦），则转换公式为 dBm=30+log(*W*)。参数 on/off 可以打开和关闭发射单元，auto 和 fixed 可以指定无线网卡是否自动选择发射功率。示例如下：

```
#iwconfig eth0 txpower 15
#iwconfig eth0 txpower 30mW
#iwconfig eth0 txpower auto
#iwconfig eth0 txpower off
```

– power：设置无线网卡的电源管理模式。period 'value'指定唤醒的周期，timeout 'value'指定进入休眠的等待时间，这两个参数之前可以加 min 和 max 修饰，这些值的单位为秒，可以附加 m 和 u 来指定毫秒和微秒。off/on 参数指定是否允许电源管理，all/unicast/multicast 参数指定允许唤醒的数据包类型。示例如下：

```
#iwconfig eth0 power period 2
#iwconfig eth0 power 500m unicast
#iwconfig eth0 power timeout 300u all
#iwconfig eth0 power off
```

```
#iwconfig eth0 power min period 2 power max period 4
```

- retry：设置无线网卡的重传机制。limit 'value'指定最大重传次数。lifetime 'value'指定最长重试时间，单位为秒，可以附带 m 和 u 来指定单位为毫秒和微秒。如果无线网卡支持自动模式，则在 limit 和 lifetime 之前还可以附加 min 和 max 来指定上下限值。示例如下：

```
#iwconfig eth0 retry 16
#iwconfig eth0 retry lifetime 300m
#iwconfig eth0 retry min limit 8
```

- key/enc[ryption]：设置无线网卡使用的加密密钥，此处为设置 WEP 模式的加密密钥。如果要使用 WPA，则需要 wpa_supplicant 工具包。密钥参数可以是 XXXX-XXXX-XXXX-XXXX 或者 XXXXXXXX 用法的十六进制数值，也可以是 s:xxxxxx 的 ASCII 字符。如果在密钥参数之前加了[index]，则只是设置该索引值对应的密钥，并不改变当前的密钥。直接指定[index]值可以设置当前使用哪一个密钥。指定 on/off 可以控制是否使用加密模式。open/restricted 指定加密模式，取决于不同的无线网卡，大多数无线网卡的 open 模式不使用加密且允许接收没有加密的数据包，而 restricted 模式使用加密。可以使用多个 key 参数，但只有最后一个生效。WEP 密钥可以是 40bit，用 10 个十六进制数字或者 5 个 ASCII 字符表示；也可以是 128bit，用 26 个十六进制数字或者 13 个 ASCII 字符表示。示例如下：

```
#iwconfig eth0 key 0123-4567-89
#iwconfig eth0 key [3] 0123-4567-89
#iwconfig eth0 key s:password [2]
#iwconfig eth0 key [2]
#iwconfig eth0 key open
#iwconfig eth0 key off
#iwconfig eth0 key restricted [3] 0123456789
#iwconfig eth0 key 01-23 key 45-67 [4] key [4]
```

- commit：提交所有的参数修改给无线网卡驱动。有些无线网卡驱动会先缓存无线网卡参数修改，然后使用这个命令来让无线网卡的参数修改生效。不过一般不需要使用该命令，因为无线网卡驱动最终都会使参数的修改生效，一般在 debug 时会用到。

说明

iwconfig 是 Linux Wireless Extensions（LWE）的用户层配置工具之一。LWE 是 Linux 下对无线网络配置的工具，包括内核的支持、用户层配置工具和驱动接口的支持 3 部分。目前很多无线网卡都支持 LWE，而且主流的 Linux 发布版本，如 Red Hat Linux 和 Ubuntu Linux 都已经附带了这个配置工具。

应用实例如下。

（1）连接无线网络。

连接到网络 MyHome（即 ESSID 为 MyHome 的网络），输入如下命令：

```
iwconfig wlan0 essid "MyHome"
```

如果网络是加密的，密码是 0123456789，则输入如下命令：

```
iwconfig wlan0 essid "MyHome" key 0123-4567-89
```

如果正常的话，则输入如下命令：

```
iwconfig wlan0
```

这样就可以看到连接正常的各项参数了。

（2）启用无线网卡。

```
ifconfig wlan0 up
```

如果是用 DHCP 获取 IP 的，那么就用 dhclient 或 dhcpcd 获取 IP。

```
dhclient wlan0
```

5.8 iw：新一代无线网络配置工具

作用：连接无线网络节点。

用法：iw[OPTIONS]{help[command]|OBJECT COMMAND}

主要参数举例说明如下。

1. 获取设备能力

用以下命令获取所有设备能力，如频段信息（2.4 GHz and 5 GHz），以及 802.11n 信息。

```
iw list
```

2. 扫描

示例如下：

```
iw dev wlan0 scan
```

3. 列出事件

示例如下：

```
iw event
```

在调试时，使用以下命令对查看 auth/assoc/deauth/disassoc frames 很有用。

```
iw event -f
```

有时候定时信息也很有用。

```
iw event -t
```

4. 获取连接状态

可以使用以下命令确定是否连接到一个 AP 和最近一次传送速率。

关联到一个非 802.11n 的信息如下：

```
iw dev wlan0 link
Connected to 04:21:b0:e8:c8:8b (on wlan0)
       SSID: attwifi
       freq: 2437
       RX: 2272 bytes (18 packets)
       TX: 232 bytes (3 packets)
       signal: -57 dBm
       tx bitrate: 36.0 MBit/s
```

关联到一个 802.11n AP 的例子：

```
iw dev wlan0 link
Connected to 68:7f:74:3b:b0:01 (on wlan0)
       SSID: tesla-5g-bcm
       freq: 5745
       RX: 30206 bytes (201 packets)
       TX: 4084 bytes (23 packets)
       signal: -31 dBm
       tx bitrate: 300.0 MBit/s MCS 15 40Mhz short GI
```

未连接到 AP 的例子：

```
iw dev wlan0 link
Not connected.
```

如果未连接到一个 AP，以上打印会出现；连接到一个 AP，则可以使用 iw connect。如果连接需求为：

```
No encryption（没有加密）
Uses WEP for encryption（使用 Wep 加密方法）
```

如果需要使用 WPA 或者 WPA2 加密连接到 AP，那么必须使用 wpa_supplicant。

5. 建立一个基本的连接

可以使用 iw 连接到以下类型的 AP：

- 未加密。
- 使用 WEP 加密。

如果用户在一个繁忙的网络中，很容易掉线，需要重发命令验证。如果不想这样，只需要使用 wpa_supplicant，它可以自动重连。

连接到 SSID 为 foo 的未加密的网络命令如下：

```
iw wlan0 connect foo
```

加入有两个 SSID 为 foo 的 AP，并且想连接的是频率为 2432 的，则可以使用以下命令：

```
iw wlan0 connect foo 2432
```

连接到使用 WEP 的 AP，可以使用如下命令：

```
iw wlan0 connect foo keys 0:abcde d:1:0011223344
```

6．获取站点统计数据

获取站点统计数据，如发送和接收字节数、最近传送比特率，可以使用以下命令：

```
$ iw dev wlan1 station dump
Station 12:34:56:78:9a:bc (on wlan0)
        inactive time:  304 ms
        rx bytes:      18816
        rx packets:     75
        tx bytes:      5386
        tx packets:     21
        signal:        -29 dBm
        tx bitrate:     54.0 Mbit/s
```

7．获取具体的统计数据

如果想获取与通信站点的统计信息，则可以使用以下命令：

```
sudo iw dev wlan1 station get <peer-MAC-address>
```

其中，<peer-MAC-address>是 AP 的 MAC 地址。

8．修改传送比特率

可以设置一个指定的比特率，例如：

```
iw wlan0 set bitrates legacy-2.4 12 18 24
```

设置 MCS 速率暂不支持。

9．设置发送功率

例如可以设置无线网络接口的发送功率。

```
iw dev <devname> set txpower <auto|fixed|limit> [<tx power in mBm>]
iw phy <phyname> set txpower <auto|fixed|limit> [<tx power in mBm>]
```

10．节能模式

开启节能模式，命令如下：

```
sudo iw dev wlan0 set power_save on
```

对于 mac80211 驱动意味着允许动态节能。

查询当前节能模式设置，命令如下：

```
iw dev wlan0 get power_save
```

11．使用 iw 添加接口

当前支持的接口模式有 monitor、managed（also station）、wds、mesh（also mp）、ibss（also adhoc）。

创建 monitor 接口，命令如下：

```
iw phy phy0 interface add moni0 type monitor
```

其中，moni0 为接口名；monitor 为模式，可替换。

当监视 802.11n 时，需要指定通道带宽（20 或 20/40MHz），对于 20/40MHz 要指定是 upper 还是 lower 通道。可通过如下命令设置：

```
iw dev <devname> set freq <freq> [HT20|HT40+|HT40-]
```

或：

```
iw phy <phyname> set freq <freq> [HT20|HT40+|HT40-]
```

也可指定信道而不是频率，命令如下：

```
iw phy <phyname> set channel <channel> [HT20|HT40+|HT40-]
iw dev <devname> set channel <channel> [HT20|HT40+|HT40-]
```

创建一个 managed 模式的接口，命令如下：

```
iw phy phy0 interface add wlan10 type managed
```

> **注意**
> 当使用 hostapd 时，接口会自动转换成 AP 模式。

12．用 iw 删除接口

```
iw dev moni0 del
```

13．用 iw 设置频点

```
iw dev wlan0 set freq 2412 [HT20|HT40+|HT40-]
```

14．用 iw 设置通道

```
iw dev wlan0 set channel 1 [HT20|HT40+|HT40-]
```

应用实例如下。

（1）查看无线网络接口名称。

```
# /sbin/iw dev
phy#0
Interface wlp0s29f7u4
    ifindex 3
    wdev 0x1
    addr 14:e6:e4:29:6c:16
    type managed
    channel 6 (2437 MHz), width: 20 MHz, center1: 2437 MHz
```

上面的例子中，无线网络接口名称是 wlp0s29f7u4 。

（2）查看驱动安装情况。

```
#  ethtool -i wlp0s29f7u4
driver: ath9k_htc
version: 3.10.0-123.6.3.el7.x86_64
firmware-version: 1.3
bus-info: 1-4:1.0
supports-statistics: yes
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no
```

上面的例子中，无线网络接口驱动是 ath9k_htc，版本是 3.10.0，固件版本是 1.3。

（3）获取 SSID。

```
# iw  wlp0s29f7u4 scan | grep SSID
  SSID: TP-LINK_BB006E
  SSID: wpf
  SSID: liuyan
  SSID: superzhang
  SSID: gehua03111406180951017
  SSID: homaki
  SSID: TP-LINK_BDEE
  SSID: D-Link_DIR-619L
  SSID: Netcore_3434F4
  SSID: TP-LINK_482DE8
  SSID: nuonuo
```

```
    SSID: TP-LINK_B614
    SSID: goodcjh12
```

说明

这里笔者使用的无线路由器 SSID 是 goodcjh12。

（4）连接一个没有安全设置的无线路由器

假设这里笔者把无线路由器 SSID 是 goodcjh12 设置为没有任何安全设置，连接方法如下。

首先打开接口。

```
# ip link set wlp0s29f7u4 up
```

然后进行连接。

```
# iw wlp0s29f7u4 connect -w Cisco1250
wlp0s29f7u4 (phy #0): [ 117.134643] wl1271: Association completed.
connected to d0:d0:fd:69:ca:70
```

下面查看一下连接状态。

```
# iw wlp0s29f7u4 link
  Connected to d0:d0:fd:69:ca:70 (on wlp0s29f7u4)
          SSID: goodcjh12
          freq: 2462
          RX: 2080 bytes (23 packets)
          TX: 344 bytes (4 packets)
          signal: -70 dBm
          tx bitrate: 54.0 MBit/s
```

（5）连接一个使用 WEP 加密的无线路由器。

假设这里笔者把无线路由器 SSID 是 goodcjh12 设置 WEP，WEP 密码是 1234567890。连接方法如下。

首先打开接口。

```
# ip link set wlp0s29f7u4 up
```

然后进行连接。

```
# iw wlp0s29f7u4 connect -w goodcjh12 key 0:1234567890
```

此处对 key 的写法略作解释。

```
key d:index:data
```

- d：default (transmit) key，默认传输密码。

- index：0～3 之间的单个数字，笔者第一次给的是 0，能获取到 DHCP IP，后面换成其他数字后运行 dhclient 获取不到 IP。
- data：必须是 5 或 13 位的 ASCII 字符，或者 10 或 26 位的十六进制数字。

最后断开连接。

```
# iw wlp0s29f7u4 disconnect
```

（6）连接一个 WPA 加密的无线路由器。

假设这里笔者把无线路由器 SSID 是 goodcjh12 设置 WPA，WPA 密码是 1234567890。连接方法如下。

首先打开接口。

```
# ip link set wlp0s29f7u4 up
```

然后进行设置。

```
#wpa_passphrase goodcjh12 》/etc/wpa_supplicant/wpa_supplicant.conf
1234567890
```

查看一下配置文件。

```
# cat /etc/wpa_supplicant.conf
# reading passphrase from stdin
network={
 ssid="goodcjh12"
 #psk="1234567890"
 psk=42e1cbd0f7fbf3824393920ea41ad6cc8528957a80a404b24b5e4461a31c820c
}
```

然后进行连接。

```
wpa_supplicant -B -D wext -i wlan0 -c /etc/wpa_supplicant.conf
```

获取无线路由器提供的 DHCP IP 地址。

```
# dhclient -r wlp0s29f7u4
 #申请前先 release
# dhclient  wlp0s29f7u4
```

已经可以连通外网。

```
[root@ www.linuxidc.com ~]# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=44 time=338 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=44 time=362 ms
^C
--- 8.8.8.8 ping statistics ---
```

```
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 338.816/350.498/362.181/11.697 ms
```

最后断开连接。

```
# iw wlp0s29f7u4 disconnect
```

5.9 hostname：显示主机名

作用：显示和设置系统主机名。
用法：hostname [选项]
主要选项如下。

- -v：显示主机名称。
- -d：显示 DNS 域名。
- -y：显示 NIS 名称。
- -f：显示 FQDN 全名 Fully Qualified Domain Name，一个域的全名。命名规则是只能使用字符，如 a～z、A～Z、0～9 和减号。
- -i：当前主机名的 IP 地址。
- -a：显示别名。
- -d,--domain：显示 DNS 域名。不要使用命令 domainname 来获得 DNS 域名，因为这会显示 NIS 域名而非 DNS 域名。可使用 dnsdomainname 替换之。
- -F,--file filename：从指定文件中读取主机名。注释（以一个`#'开头的行）可忽略。
- -n,--node：显示 DECnet 节点名。如果指定了参数（或者指定了--file name），那么 root 也可以设置一个新的节点名。
- -s,--short：显示短格式主机名。这是一个去掉第一个圆点后面部分的主机名。

说明

无论在局域网还是互联网上，每台主机都有一个 IP 地址，为了区分此台主机和彼台主机，也就是说 IP 地址就是主机的门牌号。但 IP 地址不方便记忆，所以又有了域名。主机名是用来做什么的呢？在一个局域网中，每台机器都有一个主机名，使得主机与主机之间便于区分。为每台机器设置主机名，以容易记忆的方法来相互访问。例如，在局域网中可以根据每台机器的功能来为其命名。Fedora/Redhat 或以 Fedora/Redhat 为基础打包的发行版，主机名配置文件是/etc/hosts。

应用实例如下。
（1）显示主机名。

```
#  hostname
localhost
```

此主机的主机名是 localhost，不加参数时用来显示当前操作的主机的主机名。

（2）临时设置主机名。

可以用 hostname 后接主机名，这样就可以设置当前操作的主机的主机名。例如，想把主机名设置为 linux02。

```
# hostname linux02
# hostname
linux02
```

通过 hostname 工具来设置主机名只是临时的，下次重启系统时，此主机名将不再存在。如果想修改主机名后一直有效，还是要用前面所讲的修改主机名配置文件/etc/hosts。

（3）显示主机 IP 地址。

显示当前主机名的 IP 地址，可以用-i 参数。

```
# hostname -i
192.168.1.9
```

5.10 ifup：激活网络设备

作用：激活网络设备。

用法：ifup [网络设备]

应用实例如下。

激活网卡 eth1。

```
#ifup eth1
```

说明

通常 Linux 管理员习惯使用手工配置网络设备的文件，如 etc/sysconfig/network。要使用已保存的配置来激活网络设备，请使用 ifup 命令。在 Linux 中把硬件设备当作文件来管理。Linux 网络设备在配置时被赋予别名，该别名由一个描述性的缩略词和一个编号组成。某种类型的第一个设备的编号为 0，其他设备依次被编号为 1、2、3 等。在给设备命名时将使用下面的约定。

– eth0, eth1…：以太网卡接口。它们用于大多数的以太网卡，包括许多并行端口以太网卡。
– tr0, tr1…：令牌环网卡接口。它们用于大多数的令牌环网卡。
– s10，s11…：SLIP 接口。SLIP 接口与串行线关联，关联顺序就是它们被分配给 SLIP 的

顺序。

- ppp0, ppp1…：PPP 接口。就像 SLIP 接口一样，PPP 接口一旦被转换到 PPP 模式，它就与串行线关联。

5.11 ifdown：禁用网络设备

作用：禁用网络设备。

用法：ifdown [网络设备]

应用实例如下。

禁用网卡 eth1。

```
#ifdown eth1
```

说明

通常 Linux 管理员习惯使用手工配置网络设备，如网卡 MAC 地址。修改之前要禁用网络设备，请使用 ifdown 命令，否则系统会报错。

5.12 mii-tool：调整网卡模式

作用：设置网卡的工作模式。

用法：mii-tool [-VvRrwl] [-A media,…| -F media] [interface…]

media: 100baseTx-FD, 100baseTx-HD, 10baseT-FD, 10baseT-HD, 100baseTx, 10baseT

主要选项如下。

- -V：查看网卡现在的工作模式。
- -F：表示强行修改。
- -r：恢复网卡的自适应工作模式。
- -w：显示监测链路状态。
- -A：使用高级修改模式。
- -l：表示将修改记录写入 syslog 日志文件。
- -p, --phy=addr：设置 MII 地址然后报告。
- media：可选的模式有 100baseTx-FD、100baseTx-HD、10baseT-FD、10baseT-HD 等。
- interface：代表所选择的网卡，如 eth0、eth1 等，默认为 eth0。

说明

现在的网卡大都是自适应工作模式的，在配置网卡参数时我们很少考虑它的工作模式，但是在路由器、交换机、代理服务器等通信量比较大的关键设备上，应该为它指定正确的工作模式，这样可以提高通信效率。在 Linux 环境下，可以使用系统自带的工具 mii-tool 来配置网卡的工作模式。

应用实例如下。

（1）查看 Linux 服务器网卡现在的工作模式。

```
# mii-tool
```

（2）显示 Linux 服务器网卡支持的所有以太网卡类型。

```
# mii-tool -v
eth0: negotiated 100baseTx-FD, link ok
product info: vendor 00:00:00, model 0 rev 0
basic mode:    autonegotiation enabled
basic status: autonegotiation complete, link ok
capabilities: 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
advertising:   100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
link partner: 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
```

从以上信息中可以看出，这块网卡工作在 10Mbit/s 全双工自适应模式下，“10Base Tx-FD”意为 10M Full Duplex。

（3）强制网卡工作在 100Mbit/双工模式下。

```
#mii-tool -F 100baseTx-FD eth0
```

（4）恢复网卡的自适应工作模式。

```
#mii-tool -r eth0
```

5.13 route：设置路由表

作用：route 的主要功能是管理 Linux 系统内核中的路由表。它最大的用途就是用来设定静态的路由表项，通常是在系统用 ifconfig 配置网络接口（如网卡等）后，再用它来设定主机或者一个网段的 IP 地址应该通过什么接口发送等。

用法：route[选项]

主要选项如下。

- -n：不显示域名。
- -e：显示其他信息。
- -v：显示详细信息。
- -F：显示 FIB 信息。
- -C：显示 FIB 缓存信息。
- add：增加路由。
- del：删除路由。
- -net：路由到达的是一个网络，而不是一台主机。
- -host：路由到达的是一台主机。
- -netmask Nm：指定路由的子网掩码。
- gw：指定路由的网关。

说明

在配置网络时，要为机器指定接收数据包时该包要经过的路径。在 Linux 系统中提供了一个命令 route，这个命令可以为 ifconfig 命令配置的网卡设置静态路由。

应用实例如下。

通常一个安装双网卡的 Linux 计算机具有路由器的功能，route 命令用来查看和设置 Linux 系统的路由信息，以实现与其他网络的通信。要实现两个不同的子网之间的通信，需要一台连接两个网络的路由器或者同时位于两个网络的网关来实现。在 Linux 系统中，设置路由通常是为了解决以下问题：该 Linux 系统在一个局域网中，局域网中有一个网关，能够让机器访问 Internet，那么就需要将这台机器的 IP 地址设置为 Linux 机器的默认路由。

（1）增加一个默认路由。

```
#route add -net 192.56.76.0 netmask 255.255.255.0 dev eth0
```

（2）查看路由表。

```
# route
Kernel IP routing table
Destination     Gateway     Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *           255.255.255.0   U      0       0        0 eth0
169.254.0.0     *           255.255.0.0     U     0      0          0 eth0
```

输出结果中各个字段的含义如下。

- Destination 表示路由的目标 IP 地址。
- Gateway 表示网关使用的主机名或者 IP 地址。上面输出的“*”表示没有网关。
- Genmask 表示路由的网络掩码。在把它与路由的目标地址进行比较之前，内核通过 Genmask 和数据包的 IP 地址进行按位“与”操作来设置路由。

- Flags 表示路由的标志。可用的标志及其意义：U 表示路由在启动，H 表示目标是一台主机，G 表示使用网关，R 表示对动态路由进行复位设置，D 表示动态安装路由，M 表示修改路由，!表示拒绝路由。
- Metric 表示路由的单位开销量。
- Ref 表示依赖本路由现状的其他路由数目。
- Use 表示路由表条目被使用的数目。
- Iface 表示路由所发送的包的目的网络。

通过查看这些输出信息，用户就可以方便地管理网络的路由表了。

5.14 netstat：查看网络连接

作用：用来检查 Linux 网络设备的工作状态。

用法：netstat [选项]

主要选项如下。

- -a,--all：显示所有连线中的 socket。
- -A：列出该网络类型连线中的 IP 相关地址和网络类型。
- -c,--continuous：连续显示网络状态。
- -C,--cache：显示路由器配置的快取信息。
- -e,--extend：显示网络其他相关信息。
- -F,--fib：显示 FIB。
- -g,--groups：显示多重广播功能群组组员名单。
- -h,--help：在线帮助。
- -i,--interfaces：显示网络界面信息表单。
- -l,--listening：显示监控中的服务器的 socket。
- -M,--masquerade：显示伪装的网络连线。
- -n,--numeric：直接使用 IP 地址，而不通过域名服务器。
- -N,--netlink--symbolic：显示网络硬件外围设备的符号链接名称。
- -o,--timers：显示计时器。
- -p,--programs：显示正在使用 socket 的程序识别码和程序名称。
- -r,--route：显示 Routing Table。
- -s,--statistice：显示网络工作信息统计表。
- -t,--tcp：显示 TCP 传输协议的连线状况。
- -u,--udp：显示 UDP 传输协议的连线状况。
- -v,--verbose：显示命令执行过程。
- -V,--version：显示版本信息。
- -w,--raw：显示 RAW 传输协议的连线状况。

- -x,--unix：和指定“-A unix”参数相同。
- --ip,--inet：和指定“-A inet”参数相同。

说明

无选项时, netstat 显示打开的套接字。如果不指定任何地址族，则打印出所有已配置地址族的有效套接字。

应用实例如下。

（1）简化 netstat 输出。

netstat 主要用于 Linux 查看自身的网络状况，如开启的端口、在为哪些用户服务，以及服务的状态等。此外，它还显示系统路由表、网络接口状态等。可以说，它是一个综合性的网络状态查看工具。在默认情况下，netstat 只显示已建立连接的端口。如果要显示处于监听状态的所有端口，使用-a 参数即可。但是显示结果输出往往有几百行，可以使用“netstat--inet-a”选项查看真正的网络连接。

```
[root@www root]# netstat --inet -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address       Foreign Address        State
tcp        0      0 *:exec              *:*                    LISTEN
tcp        0      0 *:netbios-ssn       *:*                    LISTEN
tcp        0      0 *:x11               *:*                    LISTEN
tcp        0      0 www.cao.com:smtp    *:*                    LISTEN
tcp        0      0 192.168.1.10:ssh   192.168.1.4:1032    ESTABLISHED
```

其中，LISTEN 表示等待接收的连接，ESTABLISHED 表示一个活跃状态的连接。

（2）使用 netstat 命令查看路由表。

Linux 命令往往具有互补性，一个命令可以通过添加适当的参数代替其他命令。用户可以根据自己的习惯完成任务。如果要查看路由表，可以使用命令“netstat -r”，它和 route 命令是相同的。

```
# netstat -r
Kernel IP routing table
Destination     Gateway      Genmask         Flags   MSS Window  irtt Iface
192.168.40.0    *             255.255.255.0   U        0 0          0 eth0
169.254.0.0     *             255.255.0.0     U        0 0          0 eth0
```

（3）查看 3306 端口（mysql）的链接数。

```
#netstat -anp |grep 3306 -c
```

5.15 ping：检测主机的连通性

作用：ping 命令用来检测主机的连通性，即用来检测主机网络接口状态。

用法：ping [选项] [host|ip]

主要选项如下。

- -d：使用 socket 的 SO_DEBUG 功能。
- -c<count>：设置完成要求回应的次数。
- -f：以高速方式来做 ping。
- -i<interval>：指定收发信息的间隔时间。
- -I<interface>：使用指定的网络界面送出数据包。
- -n：只输出数值。
- -p<pattern>：设置填满数据包的范本样式。
- -q：不显示命令执行过程，开头和结尾的相关信息除外。
- -r：忽略普通的路由表，直接将数据包送到远端主机上。
- -R：记录路由过程。
- -s<sndbuf>：设置数据包的大小。
- -t<timestamp option>：设置存活数值 TTL 的大小。
- -v：详细显示命令的执行过程。
- -w：等待指定时间后停止 ping 程序的执行。当试图检测不可达主机时此选项很有用。
- host|ip：主机名称或 IP 地址。

说明

ping 命令是使用最多的网络命令，通常使用它检测网络是否连通，它使用 ICMP（ICMP 是 Internet 控制报文协议）。但是有时会有这样的情况，我们可以用浏览器查看一个网页，但是却无法 ping 通，这是因为一些网站出于安全考虑安装了防火墙。

应用实例如下。

（1）连接远程主机 IP 地址 3 次。

```
# ping 192.168.40.11 -c 3
Pinging any-fp.wa1.b.yahoo.com [98.137.149.56] with 32 bytes of data:
PING 192.168.40.11 (192.168.40.11) 56(84) bytes of data.
64 bytes from 192.168.40.11: icmp_seq=1 ttl=64 time=0.253 ms
64 bytes from 192.168.40.11: icmp_seq=2 ttl=64 time=0.285 ms
64 bytes from 192.168.40.11: icmp_seq=3 ttl=64 time=0.276 ms
--- 192.168.40.11 ping statistics ---
```

```
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.253/0.271/0.285/0.019 ms
```

（2）连接雅虎网站 3 次，但是以 IP 地址显示。

```
# ping www.yahoo.com –n -c 3
Pinging any-fp.wa1.b.yahoo.com [98.137.149.56] with 32 bytes of data:
64 bytes from 98.137.149.56: icmp_seq=1 ttl=64 time=0.253 ms
64 bytes from 98.137.149.56: icmp_seq=2 ttl=64 time=0.285 ms
64 bytes from 98.137.149.56: icmp_seq=3 ttl=64 time=0.276 ms
--- 192.168.40.11 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.253/0.271/0.285/0.019 ms
```

5.16 minicom：设置调制解调器

作用：minicom 命令用来配置调制解调器。

用法：minicom [选项] [configuration]

主要选项如下。

- -s, --setup：进入设置模式（只有超级用户可以执行）。
- -o, --noinit：不要初始化调制解调器。
- -m, --metakey：以 Alt 键或 Meta 键作为命令键。
- -M, --metakey8：以 Alt 键或 Meta 键作为命令键。
- -l, --ansi：不会将所有的字符都转成 ASCII 码。
- -z, --statline：在终端上显示状态列。
- -8, --8bit：不要修改任何 8 位编码的字符。
- -c, --color=on/off：设置彩色模式。
- -a, --attrib=on/off：设置终端属性。
- -t, --term=TERM：设置终端类型。
- -S, --script=SCRIPT：在启动时，执行指定的脚本文件。
- -d, --dial=ENTRY：启动或直接拨号。
- -p, --ptty=TTYP：使用模拟终端机。
- -C, --capturefile=FILE：指定取文件，并在启动时开启取功能。
- -v, --version：显示版本信息并且退出。
- configuration：指定 minicom 配置文件。

说明

minicom 是一个相当受欢迎的 PPP 拨号连接命令。使用前首先检查是否安装了 minicom 软件包，可以使用命令“rpm –qa|grep minicom”。

应用实例如下。

初始化调制解调器，使用如下命令：

```
#minicom -s
```

进入交互界面，如图 5-1 所示。

图 5-1　初始化调制解调器

5.17 pppd：建立 PPP 连接

作用：pppd 命令用来在 56K 调制解调器和 PPP 服务器之间建立并维持连接、传输数据。

用法：pppd [选项]

主要选项如下。

- <device>：建立连接的设备名称。
- <speed>：连接速率。
- auth：要求远端在允许传送或接收网络封包之前先验证它自己。
- mru <n>：把 MRU（Maximum Receive Unit，最大接收单元）的值设为 n 来进行协商。pppd 将会要求远端传送不比位组更长的封包。最小的 MRU 值是 128。预设的 MRU 值是 1500。对于慢速线路上的建议值是 296。
- asyncmap：把异步（async）字符设为对照表。这个对照表描述哪些控制字符不能在串联线路上成功地接收。
- crtscts：使用硬件流量控制（RTS/CTS）来控制串列埠上的资料流。
- file：从指定文件中读取选项。
- modem：使用数据机控制线路。

说明

pppd（Point to Point Protocol Daemon）是点对点通信协议的守护进程。使用前首先检查是否安装了 PPP 软件包，可以使用命令“rpm –qa|grep ppp”。

应用实例如下。

1）用 pppd 命令建立 PPP 连接

测试拨号能否成功，然后退出。

```
#pppd debug lock Modem crtscts defaultroute /dev/ttyS1 57600&
```

在一个单独的虚拟控制台上运行下面的代码。

```
#tail-f/var/log/messages
```

上面的命令用来动态地观察各种日志信息，包括 pppd 的调试信息。如果看到下面的信息，就说明 PPP 连接成功了。

```
#Jun23 16:10:31hxf pppd[1234]:local IP address210.88.131.151
Jun23 16:10:31hxf pppd[1234]:remote IP address192.168.10.21
```

2）断开连接

当测试通过后，需要把连接断开，使用如下命令：

```
#killall pppd
```

注意

这个命令对于按时间计费的用户非常重要。因为使用内置的 PPP 拨号器时，即使不使用浏览器，也可能因为别的原因不能断线，计费仍在继续。另外，在不访问 Internet 的情况下，应及时断开调制解调器电源。

pppd 错误代码如下。

- 0：pppd 已经断开，或者已经成功建立连接后请求方又中断了。
- 1：发生了一个严重错误，例如，系统调用失败或者访问非法内存。
- 2：处理给定操作时检测到错误，例如，使用两个互斥的操作。
- 3：调用 pppd 的用户没有权限（必须是超级用户）。
- 4：内核不支持 PPP，例如，PPP 驱动没有加载或者内核编译没有包含 PPP 驱动。
- 5：pppd 收到 SIGINT、SIGTERM、SIGHUP 而中断。
- 6：不能锁定串口。
- 7：不能打开串口。
- 8：拨号脚本错误（退出状态非 0）。
- 9：参数中指定的 pty 不可用。
- 10：PPP 连接错误，也就是说在建立 PPP 连接时所有可用的网络协议都不可到达。
- 11：服务器验证失败，拒绝连接。
- 12：链路成功建立，但是由于链路空闲所以中断连接。
- 13：链路已经成功建立，但由于连接超时导致中断连接。
- 14：已经完成握手协议，连接成功。

- 15：对方没有响应，连接中断。
- 16：调制解调器挂起导致链路中断。
- 17：重复请求被检测到，导致 PPP 连接中断（已经存在一个连接请求）。
- 18：连接初始化脚本失败（返回一个非 0 值）。
- 19：验证自己失败。

5.18 pppstats：显示 PPP 连接状态

作用：pppstats 的全称是 Point to Point Protocol Status，用于显示 PPP 连接状态。

用法：pppstats [-adrv] [-c<执行次数>] [-w<间隔秒数>] [网络界面]

主要选项如下。

- -a：显示绝对统计值。
- -c<执行次数>：设置回报状况的次数。
- -d：显示相对统计值。
- -r：显示数据包压缩比率的统计值。
- -v：显示 VJTCP 文件头的压缩效率统计值。
- -w<间隔秒数>：设置显示统计信息的间隔时间。

> **说明**
> 利用 pppstats 命令可让用户得知 PPP 连接网络的相关信息。

应用实例如下。

显示 PPP 连接状态使用信息。

```
# pppstats -a ppp0
```

5.19 chat：拨号命令

作用：chat 命令用来拨号，并等待提示，根据提示输入用户名和密码等登录信息。

用法：chat [选项] script

主要选项如下。

- -r report-file：产生日志文件。
- -t timeout：设置拨号失败退出时间。
- -T phone-number：第一个拨号电话号码。

– -U phone-number2：第二个拨号电话号码。
– {-f chat-file | chat-script}：从指定文件读取脚本。

> **说明**
> chat 命令使用前首先检查是否安装了 chat 软件包，使用命令“rpm －qa|grep chat”。

5.20 traceroute：检查数据包所经过的路由

作用：显示数据包到达目的主机所经过的路由。

用法：traceroute [-dFlnrvx] [-f<存活数值>] [-g<网关>...] [-i<网络界面>] [-m<存活数值>] [-p<通信端口>] [-s<来源地址>] [-t<服务类型>] [-w<超时秒数>] [主机名称或 IP 地址] [数据包大小]

主要选项如下。

– -d：使用 socket 层级的排错功能。
– -f<存活数值>：设置第一个检测数据包的存活数值 TTL 的大小。
– -F：设置勿离断位。
– -g<网关>：设置来源路由网关，最多可设置 8 个。
– -i<网络界面>：使用指定的网络界面送出数据包。
– -I：使用 ICMP 回应取代 UDP 资料信息。
– -m<存活数值>：设置检测数据包的最大存活数值 TTL 的大小。
– -n：直接使用 IP 地址而非主机名称。
– -p<通信端口>：设置 UDP 传输协议的通信端口。
– -r：忽略普通的 Routing Table，直接将数据包送到远程主机上。
– -s<来源地址>：设置本地主机送出数据包的 IP 地址。
– -t<服务类型>：设置检测数据包的 TOS 数值。
– -v：详细显示命令的执行过程。
– -w<超时秒数>：设置等待远端主机回报的时间。
– -x：开启或关闭数据包的正确性检验。

> **说明**
> 什么是 traceroute?Internet 是目前世界上最大的计算机网络，更确切地说是网络的网络。它由遍布全球的几万个局域网和数百万台计算机组成，并通过用于异构网络的 TCP/IP 进行网间通信。在互联网中，信息的传送通过网中许多段的传输介质和设备（路由器、交换机、服务器、网关等）从一端到达另一端。每一个连接在 Internet 上的设备，

如主机、路由器、接入服务器等，一般情况下都会有一个独立的 IP 地址。通过 traceroute 我们可以知道，信息从用户的计算机到互联网另一端的主机走的是什么路径。当然每次数据包由某一同样的出发点（source）到达某一同样的目的地（destination）走的路径可能会不一样，但基本上来说大部分时间所走的路径是相同的。在 UNIX 和 Linux 系统中，我们称之为 traceroute；在 MS Windows 系统中称之为 Tracert。traceroute 通过发送小的数据包到目的设备直到其返回，来测量其需要多长时间。一条路径上的每个设备，traceroute 要测 3 次。输出结果中包括每次测试的时间（ms）和设备的名称（如有的话）及其 IP 地址。traceroute 命令让用户追踪网络数据包的路由途径，预设数据包大小是 40Bytes，用户也可另行设置。

应用实例如下。

（1）显示到一个 IP 地址的路由。

```
# traceroute 192.168.118.1
traceroute to 192.168.118.1 (192.168.118.1), 30 hops max, 40 byte packets
(192.168.118.1)  0.249 ms  0.189 ms  0.176 ms
```

（2）显示雅虎中国网站的路由。

```
#traceroute www.yahoo.com.cn
traceroute to homepage.vip.cnb.yahoo.com (202.165.102.205), 30 hops max,
38 byte packets
 1  221.217.128.1 (221.217.128.1)  18.552 ms  20.154 ms  19.222 ms
 2  bt-204-141.bta.net.cn (202.106.204.141)  19.927 ms  22.130 ms  31.461 ms
 3  202.106.36.1 (202.106.36.1)  19.323 ms  22.439 ms  20.134 ms
 4  61.148.3.97 (61.148.3.97)  18.608 ms  20.899 ms  28.814 ms
 5  202.108.46.30 (202.108.46.30)  26.150 ms  17.862 ms  28.540 ms
 6  202.108.47.130 (202.108.47.130)  24.455 ms  19.803 ms  19.446 ms
 7  61.135.148.2 (61.135.148.2)  33.238 ms  23.657 ms  34.785 ms
 8  po2.bas1.cnb.yahoo.com (202.165.96.198)  38.120 ms  19.826 ms  18.448 ms
 9  * * *
…
30  * * *
```

5.21 tracepath：追踪连接到目标地址所经过的路由

作用：追踪连接到目标地址所经过的路由。

用法：tracepath [-n] [-b] [-l <len>] [-p port] <destination>

主要选项如下。

- destination：指定追踪路由信息的目的主机。
- -p port：指定使用的 UDP 端口号。
- -n：不查找 DNS 主机以节省时间，不查寻主机名，仅仅给出 IP 地址值。
- -b：不查寻主机名和 IP 地址。
- -l：设置初始包的大小。如果不设置，则报文的大小为 65 535。

应用实例如下。

```
# tracepath www.google.cn
1: 192.168.0.3 (192.168.0.3)          0.709ms pmtu 1500
1: 192.168.0.1 (192.168.0.1)          0.908ms
2: 192.168.0.1 (192.168.0.1)          asymm 1  0.199ms pmtu 1480
3: 222.247.28.49 (222.247.28.49)      64.933ms
4: 61.137.2.169 (61.137.2.169)        91.804ms
5: 202.97.46.29 (202.97.46.29)        137.155ms
6: 202.97.35.173 (202.97.35.173)      199.915ms
7: 202.97.53.130 (202.97.53.130)      asymm 11 243.512ms
8: no reply
```

5.22 rcp：远程复制

作用：rcp 命令可以远程复制文件或目录。

用法：rcp [-pr] [源文件或目录] [目标文件或目录]或 rcp [-pr] [源文件或目录...] [目标文件]

主要选项如下。

- -r：递归地把源目录中的所有内容复制到目的目录中。要使用这个选项，目的必须是一个目录。
- -p：试图保留源文件的修改时间和模式，忽略 umask。
- -k：请求 rcp 获得在指定区域内的远程主机的 Kerberos 许可，而不是获得由 krb_relmofhost 确定的远程主机区域内的远程主机的 Kerberos 许可。
- -x：为传送的所有数据打开 DES 加密。这会影响响应时间和 CPU 利用率，但是可以提高安全性。如果在文件名中指定的路径不是完整的路径名，那么这个路径被解释为相对远程机上同名用户的主目录。如果没有给出远程用户名，就使用当前用户名。如果远程机上的路径包含特殊 Shell 字符，需要用反斜线（\\）、双引号（"）或单引号（'）括起来，使所有的 Shell 元字符都能被远程地解释。需要说明的是，rcp 不提示输入口令，它通过 rsh 命令来执行复制操作。
- directory：每个文件或目录参数既可以是远程文件名，也可以是本地文件名。远程文件名具有如下形式：rname@rhost:path，其中 rname 是远程用户名，rhost 是远程计算机名，path 是这个文件的路径。

说明

rcp 命令用于远程复制文件或目录。如果同时指定两个以上的文件或目录，且最后的目的地是一个已经存在的目录，则它会把前面指定的所有文件或目录复制到该目录中。

应用实例如下。

（1）将当前目录下的 test1 复制到名为 webserver1 的远程系统。

```
# rcp test1 webserver1:/home/root/test3
```

在这种情况下，test1 被复制到远程子目录 test3 下，名称仍为 test1 。如果仅提供了远程主机名，rcp 将把 test1 复制到远程主目录下，名称仍为 test1 。

（2）从远程系统复制目录。

要将远程目录及其所有文件和子目录复制到本地目录，请在以下语法中使用 rcp 和 -r（递归）选项。

```
# rcp –r remote_hostname:remote_dir local_dir
```

要将名为 work 的远程目录复制到当前目录，请输入以下内容：

```
# rcp –r webserver1:/home/root/work .
```

“.”表示当前目录。将在此目录下创建 work 目录。

5.23 tcpdump：网络数据分析器

作用：分析网络传输数据。

用法：tcpdump [选项] [-c<数据包数目>] [-dd] [-ddd] [-F<表达文件>] [-i<网络界面>] [-r<数据包文件>] [-s<数据包大小>] [-tt] [-T<数据包类型>] [-vv] [-w<数据包文件>] [输出数据栏位]

主要选项如下。

- -a：尝试将网络和广播地址转换成名称。
- -c<数据包数目>：收到指定的数据包数目后，就停止进行倾倒操作。
- -d：把编译过的数据包编码转换成可阅读的用法，并倾倒到标准输出。
- -dd：把编译过的数据包编码转换成 C 语言的用法，并倾倒到标准输出。
- -ddd：把编译过的数据包编码转换成十进制数字的用法，并倾倒到标准输出。
- -e：在每列倾倒资料上显示连接层级的文件头。
- -f：用数字显示网际网络地址。
- -F<表达文件>：指定内含表达方式的文件。
- -i<网络界面>：使用指定的网络界面送出数据包。
- -l：使用标准输出列的缓冲区。

- -n：不把主机的网络地址转换成名字。
- -N：不列出域名。
- -O：不将数据包编码最佳化。
- -p：不让网络界面进入混杂模式。
- -q：快速输出，仅列出少数的传输协议信息。
- -r<数据包文件>：从指定的文件读取数据包数据。
- -s<数据包大小>：设置每个数据包的大小。
- -S：用绝对而非相对数值列出 TCP 关联数。
- -t：在每列倾倒资料上不显示时间戳记。
- -tt：在每列倾倒资料上显示未经用法化的时间戳记。
- -T<数据包类型>：强制将表达方式所指定的数据包转译成设置的数据包类型。
- -v：详细显示命令执行过程。
- -vv：更详细地显示命令执行过程。
- -x：用十六进制字码列出数据包资料。
- -w<数据包文件>：把数据包数据写入指定的文件。

> **说明**
>
> 执行 tcpdump 命令可列出经过指定网络界面的数据包文件头。在 Linux 操作系统中，使用者必须是系统管理员。

应用实例如下。

（1）过滤掉除 ARP 请求和应答的所有通信数据。

```
#tcpdump arp
```

该命令只对 ARP 的请求和应答信息进行截获。在 tcpdump 的输出信息中，请求是“arp who-has”这样的条目，而应答则是“arp reply”这样的条目。如果要做更多的处理，如从指定的网络接口截获 5 个 ARP 数据包，并且不将网络地址转换成主机名，则可以使用如下命令：

```
#tcpdump arp -i eth0 -c 5 －n
```

（2）截获 IP 地址为“192.185.10.57”的计算机所有收到和发出的数据包。

```
# tcpdump host 192.185.10.57
```

（3）截获在 IP 地址为“192.185.10.57”的计算机和 IP 地址为“192.185.10.55”的计算机或 IP 地址为“192.185.10.53”的计算机之间传递的数据包。

```
# tcpdump host 192.185.10.57 and \ >\(192.185.10.55or 192.185.10.53\)
```

需要注意的是，在使用布尔运算符构成组合表达式时，有时需要使用括号来表达复杂的逻辑关系。如果要在命令行中使用括号，一定要用转义字符（\）对括号进行转义，否则命令行解

释器将给出语法错误的提示。

（4）截获除 IP 地址为“192.185.10.57”和“192.185.10.58”的计算机外所有其他主机之间通信的 IP 数据包。

```
# tcpdump ip host 192.185.10.57 and ! 192.185.10.58
```

（5）截获 IP 地址为“192.185.10.57”的计算机接收或发出的 FTP（端口号为 21）数据包。

```
# tcpdump tcp port 21 host 192.185.10.57
```

（6）截取发往本机的所有 ICMP 包。

如果怀疑系统正受到拒绝服务（DoS）攻击，网络管理员可以通过截获发往本机的所有 ICMP 包，来确定目前是否有大量的 ping 指令流向服务器，此时就可以使用下面的命令：

```
# tcpdump icmp -n -i eth0
```

下面介绍 tcpdump 的输出结果。

在对网络中的数据包进行过滤后，tcpdump 的输出结果中包含网络管理员关心的网络状态信息。由于 tcpdump 只是一个命令行方式的嗅探器，因而其输出结果不是很直观。下面以几种典型的输出信息为例，介绍如何对 tcpdump 的输出结果进行分析。

1）ARP 包的输出信息

“tiger”是一台装有 Linux 系统的主机，其 MAC 地址是 00:D0:59:BF:DA:06；“mag”是一台装有 SCO UNIX 的工作站，其 MAC 地址是 08:90:B0:2F:AF:46。若使用“tcpdump arp -c 2”命令截获 ARP 数据包，得到的输出结果可能如下：

```
20:42:22.713502 eth0 > arp who-has mag tell tiger
(00:d0:59:bf:da:06)
20:42:22.713907 eth0 < arp reply mag is-at 08:90:b0:2f:af:46
(00:d0:59:bf:da:06)
```

在输出的信息中，“20:42:22”为截获数据包的时间，“713502”和“713907”为毫秒数，“eth0 >”表明从主机发出该数据包，“eth0 <”表明从主机接收该数据包，“arp”表明该数据包是 ARP 请求，“who-has mag tell tiger”表明是主机“tiger”请求主机“mag”的 MAC 地址，“00:d0:59:bf:da:06”是主机“tiger”的 MAC 地址，“reply mag is-at”表明主机“mag”响应主机“tiger”的 ARP 请求，“08:90:b0:2f:af:46”是主机“mag”的 MAC 地址。

2）数据链路层头信息

“cjh”是一台装有 Linux 系统的主机，其 MAC 地址是 00:0c:29:50:2f:94；“c1”是一台装有 Windows XP 系统的工作站，其 MAC 地址是 00:50:56:c0:00:01，使用“tcpdump -e host cjh -c 1”命令截获主机“cjh”所有发出和收到的数据包一次，并在输出结果中包含数据链路层的头部信息。

```
# tcpdump -e host cjh -c 1
```

上述命令的输出结果如下：

```
19:27:08.265622 00:0c:29:50:2f:94 (oui c1) > 00:50:56:c0:00:01 (oui c1),
```

```
ethertype IPv4 (0x0800), length 170: cjh.ssh > 192.168.40.11.2189: P
2739737068:2739737184(116) ack 2411887553 win 1638
```

在输出的信息中，“19:27:08”为截获数据包的时间；“265622”是毫秒数；“>”表示从网络接口发送数据包；“00:0c:29:50:2f:94”是主机 cjh 的 MAC 地址，指明发送该数据包的源主机为“cjh”;“00:50:56:c0:00:01”是主机 c1 的 MAC 地址，指明该数据包发送的目标主机为“c1”;“IPv4”表明该数据包是 IP 数据包；“170”是数据包的长度；cjh.ssh 表示 ssh 连接；“ack 2411887553”表示对序列号为 2411887553 的包进行确认；“win 1638”则指明发送窗口的大小为 1638。

3）TCP 包的输出信息

用 tcpdump 截获的 TCP 包的一般输出格式如下：

```
src > dst: flags data-sequno ack window urgent options
```

“src > dst:”表明从源地址到目的地址；“flags”是 TCP 包中的标志信息，包括 S（SYN）标志、F（FIN）标志、P（PUSH）标志、R（RST）标志和“.”（没有标志）；“data-sequno”是数据包中的数据序列号；“ack”是下次期望的数据序列号；“window”是接收缓存的窗口大小；“urgent”表明数据包中是否有紧急指针；“options”是可能的选项值。

4）UDP 包的输出信息

用 tcpdump 截获的 UDP 包的一般输出格式如下：

```
src.port1 > dst.port2: udp lenth
```

UDP 中包含的信息很简单。上面的输出结果表明，从主机“src”的“port1”端口发出的一个 UDP 数据包被送到主机“dst”的“port2”端口，数据包的类型是 UDP，其长度为“lenth”。

5.24 ipcalc：IP 地址计算器

作用：使用 ipcalc，只要输入 IP 地址和子网掩码，就可以换算出子网络号、广播地址、主机名称、CIDR 等。

用法：ipcalc [选项]

主要选项如下。

- -b, --broadcast：显示指定 IP 地址和子网掩码的广播地址。
- -h, --hostname：显示指定 IP 地址的主机名。
- -m, --netmask：显示指定 IP 地址的子网掩码——特指默认。
- -n, --network：显示指定 IP 地址的网络地址。
- -p, --prefix：显示网络前缀。
- -s, --silent：不显示错误信息。

应用实例如下。

1）换算出子网络号、广播地址、主机名称信息

计算出 192.168.10. 192/26 的子网掩码、网络号、广播地址、各子网范围。

```
# ipcalc -bmn 192.168.10.192/26
NETMASK=255.255.255.192
BROADCAST=192.168.10.255
NETWORK=192.168.10.192
```

由以上操作可知，子网掩码为 255.255.255.192。也可获得 4 个子网的网段地址、网络号和广播地址。

2）进行 CIDR

ipcalc 还有一个功能是进行 CIDR（Classless Inter-Domain Routing，无类域间路由），即取消 IP 地址的分类结构，将多个 C 类地址块聚合在一起生成一个更大的网络，以包含更多的主机。CIDR 支持路由聚合，能够将路由表中的许多路由条目合并为更少的数目，因此可以限制路由器中路由表的增大，减少路由通告。同时，CIDR 有助于 IPv4 地址的充分利用。

例如，将 192.168.8.0～192.168.15.0 这组 C 类地址聚合为一个网络。

```
$ipcalc -mn 192.168.8.0 192.168.15.0
NETMASK=255.255.255.0
NETWORK=192.168.8.0
```

即 192.168.8.0/21。

5.25 netreport：监视网络状态

作用：netreport 命令在网络接口发生变化时发送 SIGIO 信号。

用法：netreport [选项]

主要选项如下。

-r：删除调用 netreport 进程的请求。

应用实例如下。

监视网络状态，删除调用 netreport 进程的请求。

```
# netreport –r
```

5.26 ip：网络集成命令工具

作用：ip 命令是 iproute2 软件包里面的一个强大的网络配置工具，它能够替代一些传统的网络管理工具，例如 ifconfig、route 等。几乎所有的 Linux 发行版本都支持 ip 命令。

用法：ip [OPTIONS] OBJECT [COMMAND [ARGUMENTS]]

OPTIONS 是一些修改 IP 行为者改变其输出的选项。所有的选项都以“-”字符开头，分为长、短两种形式。目前，ip 命令支持的 OPTIONS 选项及其含义如表 5-1 所示。

表 5-1　OPTIONS 选项及其含义

选　项	含　义
-V,-Version	打印 IP 的版本并退出
-s,-stats,-statistics	输出更为详尽的信息。如果这个选项出现两次或多次，输出的信息将更为详尽
-f,-family	这个选项后面接协议种类，包括 inet、inet6 或 link，强调使用的协议种类。如果没有足够的信息告诉 ip 命令使用哪种协议，ip 命令就会使用默认值 inet 或 any、link 比较特殊，它表示不涉及任何网络协议
-4	是-family inet 的简写
-6	是-family inet6 的简写
-0	是-family link 的简写
-o,-oneline	对每行记录都使用单行输出，回行用字符代替。如果需要使用 wc、grep 等工具处理 ip 命令的输出，会用到这个选项
-r,-resolve	查询域名解析系统，用获得的主机名代替主机 IP 地址

OBJECT 是管理者获取信息的对象。目前 ip 命令认识的对象（OBJECT 选项）如表 5-2 所示。

表 5-2　OBJECT 选项

选　项	含　义
address	一个设备的协议（IP 或者 IPv6）地址
neighbour	ARP 或者 NDISC 缓冲区条目
route	路由表条目
rule	路由策略数据库中的规则
maddress	多播地址
mroute	多播路由缓冲区条目
tunnel	IP 上的通道

另外，所有的对象名都可以简写，例如 address 可以简写为 addr，甚至可以简写为 a。

COMMAND 设置针对指定对象执行的操作，它和对象的类型有关。一般情况下，ip 命令支持对象的增加（add）、删除（delete）和展示（show 或者 list）。有些对象不支持所有这些操作，或者其他的一些命令。对于所有的对象，用户可以使用 help 命令获得帮助。这个命令会列出这个对象支持的命令和参数的语法。如果没有指定对象的操作命令，ip 命令会使用默认的命令。一般情况下，默认命令是 list。如果对象不能列出，就会执行 help 命令。

ARGUMENTS 是命令的一些参数，它们依赖于对象和命令。ip 命令支持两种类型的参数：flag 和 parameter。flag 由一个关键词组成；parameter 由一个关键词加一个数值组成。为了方便，每个命令都有一个可以忽略的默认参数。例如，参数 dev 是 ip link 命令的默认参数，因此 ip link ls eth0 等于 ip link ls dev eth0。我们将在后面的章节中详细介绍每个命令的使用方法，命令的默认参数将使用 default 标出。

应用实例如下。

（1）添加 IP 地址 192.168.1.1/24 到 eth0 网卡上。

```
#ip addr add 192.168.1.1/24 dev eth0
```

（2）丢弃源地址属于 192.168.2.0/24 网络的所有数据报。

```
#ip rule add from 192.168.2.0/24 prio 32777 reject
```

（3）使用-statistics 选项打印出网络接口的统计信息。

```
ip -s link ls eth0
3: eth0: mtu 1500 qdisc cbq qlen 100
link/ether 00:a0:cc:66:18:78 brd ff:ff:ff:ff:ff:ff
RX: bytes packets errors dropped overrun mcast
2449949362 2786187 0 0 0 0
TX: bytes packets errors dropped carrier collsns
178558497 1783946 332 0 332 35172
```

RX 和 TX 分别是接收和发送统计信息的开头，得到的统计信息如下。

- bytes：网络接口发送或者收到的字节数。如果字节数超过数据类型能够表示的最大数值，就会造成回卷。因此，如果想连续监视这个指标，需要一个用户空间的监控进程周期性地保存这个数据。
- packets：网络接口收到或者发送的数据包个数。
- errors：发生错误的次数。
- dropped：由于系统资源限制而丢弃数据包的数量。
- overrun：由于发生堵塞，收到的数据包被丢弃的数量。如果接口发生堵塞，就意味着内核或机器太慢，无法处理收到的数据。
- mcast：收到的多播数据包数量，只有很少的设备支持这个选项。
- carrier：连接介质出现故障的次数，例如，网线接触不好。
- collsns：以太网类型介质发生冲突的事件次数。
- compressed：压缩数据包的总数。这个指标只适用于使用 VJ 头压缩的网络接口。

5.27 pppoe-setup：设置 ADSL

作用：pppoe 系列命令用于设置 ADSL 连接，它是 Shell 脚本。

用法：该命令无参数。

说明

adsl 设备管理命令包括一组命令。首先检查是否安装了 PPPoE 软件，并且版本要高于 3.5-1。如果版本低于这个版本，先升级软件版本。升级命令如下：

```
#rpm Uvh rp—pppoe—3.5-3.i386.rpm
```

系统产生如下 4 个命令。

- pppoe--setup：配置 DNS、用户账号、密码、其他选项。
- pppoe--start：使用 ADSL 连接互联网命令。
- pppoe-stop：断开 ADSL 和互联网连接命令。
- pppoe--status：检测 ADSL 连接状态命令。

应用实例如下。

配置 ADSL 具体安装过程一共 9 步，如图 5-2 和图 5-3 所示，简单介绍如下。

① 添加输入 ADSL 账号的登录账户名称，按 Enter 键继续。

② 选择 ADSL 设备所连接的网卡，笔者使用的是 eht0，按 Enter 键继续。

③ 选择 ADSL 断线后是否自动拨号，包月用户选择“yes”，其他用户选择“no”，按 Enter 键继续。

④ 输入 ISP 的静态主 DNS 地址和辅助 DNS 地址，按 Enter 键继续。

⑤ 输入 ADSL 账号的密码，然后重复输入一次，输入完成后按 Enter 键继续。

⑥ 选择是否允许普通用户开启、关闭 ADSL 连接，默认为 yes，按 Enter 键继续。

①～⑥操作过程如图 5-2 所示。

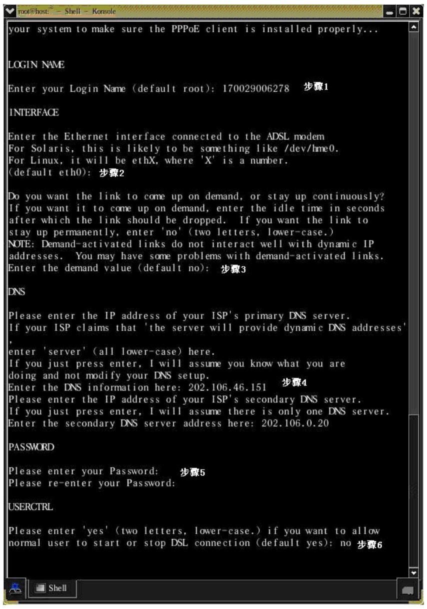


图 5-2　ADSL 安装图解 1

⑦ 选择防火墙等级，选择 0，输入完成后按 Enter 键继续。

⑧ 选择是否在启动的时候就拨号连接，默认为 no（包月用户选择“yes”），按 Enter 键继续。

⑨ 下面会显示输入信息的一个摘要，确认后选择“y”完成配置。

⑦～⑨操作过程如图 5-3 所示。

如果出现如图 5-4 所示的界面，表示 ADSL 在 Linux 中已配置完成。

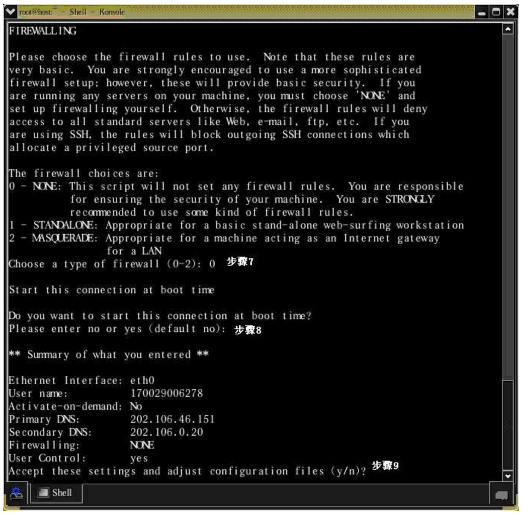


图 5-3 ADSL 安装图解 2

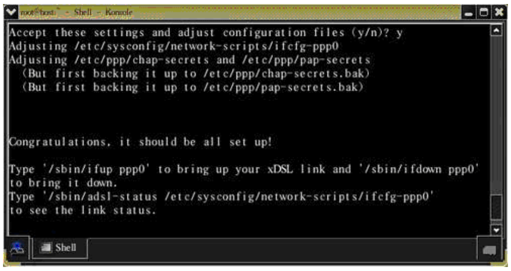


图 5-4 ADSL 在 Linux 中配置完成

5.28 pppoe-start：激活 ADSL 连接

作用：pppoe-start 命令用来激活 ADSL 连接。

用法：pppoe-start [config_file]

pppoe-start interface user [config_file]

主要选项如下。

- config_file：ADSL 配置文件。
- interface：网络接口。
- user：ADSL 用户名。

应用实例如下。

建立 ADSL 连接。

```
# pppoe-start –I eth0
```

到目前为止 Linux 计算机已经可以通过 ADSL 上网了。使用命令查看 ppp0 连接情况，如图 5-5 所示。

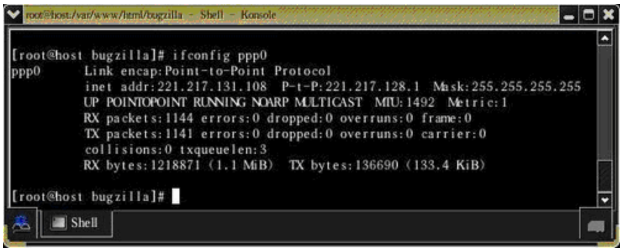


图 5-5　ppp0 连接情况

5.29 pppoe-stop：断开 ADSL 连接

作用：pppoe-stop 命令用来断开 ADSL 连接。

用法：pppoe-stop [config_file]

主要选项如下。

config_file：ADSL 配置文件。

应用实例如下。

断开 ADSL 连接。

```
# pppoe-stop
```

5.30 pppoe-status：检测 ADSL 连接状态

作用：pppoe-status 是检测 ADSL 连接状态的命令。

用法：pppoe-status [config_file]

主要选项如下。

config_file：ADSL 配置文件。

应用实例如下。

查看 ADSL 连接状态。

```
# pppoe-status /etc/sysconfig/network-scripts/ifcfg-ppp0
pppoe-status: Link is down
```

> **注意**
>
> 这个命令对于按时间计费的用户非常重要。因为使用内置的 PPPoE 拨号器时，即使不使用浏览器，也可能因为别的原因不能断线，计费仍在继续。另外应在不访问 Internet 的情况下，及时断开 ADSL 电源。

5.31 wget：下载文件

作用：wget 是 Linux 环境下用于从互联网上提取文件的工具。

用法：wget [选项]... [URL]...

URL 地址格式可以使用如下格式的 URL：

```
http://host[:port]/path
```

例如：

```
http://fly.cc.fer.hr/
    ftp://ftp.xemacs.org/pub/xemacs/xemacs-19.14.tar.gz
    ftp://username:password@host/dir/file
```

在最后一种形式中，以 URL 编码形式为 FTP 主机提供了用户名和密码（当然，也可以使用参数提供该信息）。

主要选项如下。

1）启动类参数

这一类参数主要提供软件的一些基本信息。

– -V,--version：显示软件版本号然后退出。

– -h,--help：显示软件帮助信息。

– -e,--execute=COMMAND：执行一个“.wgetrc”命令。

以上每个功能都有长、短两个参数，长、短参数功能一样，都可以使用。需要注意的是，这里的-e 参数是执行一个.wgettrc 命令，.wgettrc 命令其实是一个参数列表，直接将软件需要的参数写在一起就可以了。

2）文件处理参数

这类参数定义软件 Log 文件的输出方式等。

– -o,--output-file=FILE：将软件输出信息保存到文件。

– -a,--append-output=FILE：将软件输出信息追加到文件。

– -d,--debug：显示输出信息。

– -q,--quiet：不显示输出信息。

– -i,--input-file=FILE：从文件中取得 URL。

– -F,--force-html：把输入文件当成 HTML 文件。

– -B,--base=URL：解析与 URL 相关的 HTML 输入文件（由-i -F 选项指定）。

3）下载参数

下载参数定义下载重复次数、保存文件名等。

– -t,--tries=NUMBER：指定下载次数（0 表示无穷次）。

– -O--output-document=FILE：下载文件保存为别的文件名。

- -nc,--no-clobber：不要覆盖已经存在的文件。
- -N,--timestamping：只下载比本地新的文件。
- -T,--timeout=SECONDS：设置超时时间。
- -Y,--代理=on/off：关闭代理。

4）目录参数

目录参数主要设置下载文件保存目录与原来文件（服务器文件）的目录对应关系。

- -nd--no-directories：不建立目录。
- -x,--force-directories：强制建立目录。

5）HTTP 参数

HTTP 参数设置一些与 HTTP 下载有关的属性。

- --http-user=USER：设置 HTTP 用户。
- --http-passwd=PASS：设置 HTTP 密码。
- --代理-user=USER：设置代理用户。
- --代理-passwd=PASS：设置代理密码。

以上参数主要设置 HTTP 和代理的用户、密码。

6）递归参数设置

在下载一个网站或者网站的一个目录时，我们需要知道下载的层次，下面这些参数就可以设置。

- -r,--recursive：下载整个网站、目录（小心使用）。
- -l,--level=NUMBER：下载层次。

7）递归允许与拒绝选项参数

下载一个网站的时候，为了尽量快，有些文件可以选择下载，比如图片和声音，也可以进行设置。

- -A,--accept=LIST：可以接受的文件类型。
- -R,--reject=LIST：拒绝接受的文件类型。
- -D,--domains=LIST：可以接受的域名。
- --exclude-domains=LIST：拒绝的域名。
- -L,--relative：下载关联链接。
- --follow-ftp：只下载 FTP 链接。
- -H,--span-hosts：可以下载外面的主机。
- -I,--include-directories=LIST：允许的目录。
- -X,--exclude-directories=LIST：拒绝的目录。

说明

wget 是 Linux 环境下用于从 World Wide Web 上提取文件的工具，这是一个 GPL 许可证下的自由软件，其作者为 Hrvoje Niksic（hniksic@srce.hr）。wget 支持 HTTP 和 FTP 协议，支持代理服务器和断点续传功能，能够自动递归远程主机的目录，找到合乎条件的文件并将其下载到本地硬盘上。如果有必要，wget 将恰当地转换页面中的超链接以在本地生成可浏览的镜像。由于没有交互式界面，wget 可在后台运行，截获并忽略 HANGUP 信号，因此在用户退出登录以后，仍可继续运行。通常，wget 用于批量地下载 Internet 网站上的文件，或制作远程网站的镜像。多数 Linux 发行版本都默认包含这个工具。如果没有安装，可在 http://www.gnu.org/software/wget/wget.html 下载最新版本，编译安装。

应用实例如下。

（1）镜像下载一个网站。

```
#wget -r www.redhat.com
```

（2）镜像下载一个网站下的某个目录。

```
#wget -r www.redhat.com/mirrors/LDP
```

（3）将在本地硬盘建立 http://oneweb.com.cn/的镜像，镜像文件存入当前目录下一个名为 oneweb.com.cn 的子目录中。

```
#wget -m -l4 -t0 http://oneweb.com.cn/
```

（4）用 wget 下载 YouTube 网站上的文件。

```
wget -q -O - http://www.youtube.com/watch?v=5kIs9H0WFus | \
grep watch_fullscreen | awk -F '&' '{print $3}' | grep -v "^$"
```

执行结果如下：

```
t=OEgsToPDskIsud8tVh-F0hAMNp6jr43b。
wget -O youtube.flv \
http://www.youtube.com/get_video?video_id=5kIs9H0WFus&t=OEgsToPDskIsud8tVh-F0hAMNp6jr43b
```

下面介绍几个有用的 wget 小技巧，可以让用户更加高效而灵活地使用 wget。

```
$ wget -r -np -nd http://example.com/packages/
```

这条命令可以下载 http://example.com 网站上 packages 目录中的所有文件。其中，-np 的作用是不遍历父目录，-nd 表示不在本机重新创建目录结构。

```
$ wget -r -np -nd --accept=iso http://example.com/centos-5/i386/
```

与上一条命令相似，但多了一个--accept=iso 选项，这指示 wget 仅下载 i386 目录中所有扩展名为.iso 的文件。也可以指定多个扩展名，只需用逗号分隔即可。

```
$ wget -i filename.txt
```

此命令常用于批量下载的情形，把所有需要下载文件的地址放到 filename.txt 中，然后 wget 就会自动下载所有文件了。

```
$ wget -c http://example.com/really-big-file.iso
```

这里所指定的-c 选项的作用为断点续传。

```
$ wget -m -k (-H) http://www.example.com/
```

该命令可用来镜像一个网站，wget 将对链接进行转换。如果网站中的图像放在另外的站点，那么可以使用-H 选项。

5.32　ngrep：监控网络接口

作用：ngrep 是 grep（在文本中搜索字符串的工具）的网络版。

用法：ngrep　[选项]…

主要选项如下。

- -q：静默模式，如果没有此开关，未匹配的数据包都以“#”显示。
- -e：显示空数据包。
- -i：忽略大小写。
- -v：反转匹配。
- -x：以十六进制格式显示。
- -X：以十六进制格式匹配。
- -w：整字匹配。
- -p：不使用混杂模式。
- -t：在每个匹配的包之前显示时间戳。
- -T：显示上一个匹配的数据包之间的时间间隔。
- -M：仅进行单行匹配。
- -I：从文件中读取数据进行匹配。
- -O：将匹配的数据保存到文件。
- -n：仅捕获指定数目的数据包进行查看。
- -A：匹配到数据包后导出指定数目的数据包。
- -W：设置显示格式 byline 将解析包中的换行符。
- -c：强制显示列的宽度。
- -F：使用文件中定义的 BPF（Berkeley Packet Filter）。
- -N：显示由 IANA 定义的子协议号。

– -d：使用哪个网卡，可以用-L 选项查询。
– -L：查询网卡接口。

说明

ngrep 是 grep（在文本中搜索字符串的工具）的网络版，它力求更多的 grep 特征，用于搜寻指定的数据包。正是由于安装 ngrep 需用到 libpcap 库，所以它支持大量的操作系统和网络协议。它当前认可 IPv4/6、TCP、UDP、ICMPv4/6、IGMP 和原始交叉以太网、PPP、SLIP、FDDI、令牌网和空端口，并以与更为普遍的分组过滤工具相同的方式了解 BPF 过滤逻辑，如 tcpdump。下载和安装方式见应用实例如下。ngrep 下载地址为 http://ngrep.sourceforge.net/。

应用实例如下。

（1）如下命令让 ngrep 为 TCP 报文监控 eth0 的 80 接口，并且只包括“it168”字符串，这样就可以获得所有该子网中访问 it168 的相关信息。

```
#ngrep -qd eth0 'it168' tcp port 80
```

（2）如下命令让 ngrep 为 TCP 报文监控 eth0 的 21 接口，并且只包括“USER|PASS”字符串。我们知道 21 端口是用来进行 FTP 连接的端口，这样就可以获得用于 FTP 连接的用户名和密码。

```
#ngrep -qd eth0 ‘USER|PASS’ tcp port 21
```

（3）查看本机和 Google 交互的数据。

```
$ngrep -qd rl0 （同时在另一控制台$mazilla http://www.google.com)
U 10.178.41.31:1054 ->; 211.98.2.4:53
.............www.google.com.....
T 216.239.57.104:80 ->; 10.178.41.31:1029 [AFP]
HTTP/1.0 302 Found..Location:
http://www.google.com/intl/zh-CN/..Set-Cookie
Type: text/html..Server: GWS/2.1..Content-length: 163..Date: Sat, 08 May 20
04 13:29:26 GMT..Connection: Keep-Alive....;;...;;..
```

该命令的输出说明本机和 Google 交互的数据。

```
$ngrep -iw 'lynx'
interface: rl0 (10.178.41.0/255.255.255.0)
match: ((^lynx\W)|(\Wlynx$)|(\Wlynx\W))
T 10.178.41.31:1050 ->; 64.233.161.99:80 [AP]
GET / HTTP/1.0 Accept-Language: en..User-Agent: Lynx/2.8.4rel.1 l
ibwww-FM/2.14 SSL-MM/1.4.1 OpenSSL/0.9.7c....
```

匹配进出数据包中包括 lynx（参数“i”忽略大小写）的信息。

（4）看看 IP 地址为 10.178.41.5 的同事在做什么。

```
$ngrep host 10.178.41.5
interface: rl0 (10.178.0.0/255.255.0.0)
filter: ip and ( host 10.178.41.5 )
T 10.178.41.5:2035 ->; 61.153.48.247:80 [AP]
GET /show.asp HTTP/1.1..Host: www.music9999.com.
.Connection: Keep-Alive..Referer: http://www.music9999.com/....
T 219.133.40.79:7204 ->; 10.178.41.5:1115 [AP]
.(OICQ_2280***30.OICQ_33****314.1.0..5m..........
```

屏幕显示在浏览 music9999.com，同时用 QQ 和朋友聊天。

（5）显示除 Telnet 的数据包。

```
#ngrep -v '' port 23
```

-v 意为反转。

（6）显示网卡所有的数据包。

```
#ngrep -d eth0 ''
```

5.33 lsof：查看打开的文件

作用：lsof 全名是 list opened files，也就是列举系统中已经被打开的文件。

用法：lsof [选项]

主要选项如下。

- -c w：显示以 w 开头命令的已打开文件的信息。
- -p PID：显示指定 PID 已打开文件的信息。
- +d dir：依照文件夹 dir 来搜寻，但不会打开子目录。
- +D dir：打开 dir 文件夹及其子目录搜寻。
- -d s：以 FD 列的信息进行匹配，可使用 3～10，表示范围，3 和 10 表示某些值。
- -u：显示某用户的已经打开的文件（或该用户执行程序已经打开的文件）。
- -i：监听指定的协议、端口、主机等的网络信息，还可以使用一些参数控制显示结果。
- -l：禁止将 userID 转换为登录名称，即显示 UID。
- -n：禁止将 IP 地址转换为 hostname 主机文件。
- -P：不显示端口名称。
- -g s：从 PGID 列进行匹配。
- +f：所有路径参数都必须是文件系统，否则不能执行。
- -f：所有路径参数都将作为普通的文件。

- +L/-L：打开或关闭文件的连接数计算。当+L 没有指定时，所有的连接数都会显示（默认）；若+L 后指定数字，则只有连接数小于该数字的信息会显示；连接数会显示在 NLINK 列。
- -t：仅打印进程，方便 Shell 脚本调用。
- -F：指定输出哪个列，可通过 lsof -F?查看。
- -r：不断执行 lsof 命令，默认每 15 秒执行一次。

lsof 命令通常的输出格式为：

```
COMMAND   PID PPID PGID USER  FD   TYPE   DEVICE   SIZE    NODE NAME
```

常见字段包括如下几个。

（1）COMMAND：默认以 9 个字符长度显示命令名称。可使用+c 参数指定显示的宽度；若+c 后跟的参数为零，则显示命令的全名。

（2）PID：进程的 ID。

（3）PPID：父进程的 ID，默认不显示，使用-R 参数可打开。

（4）PGID：进程组的 ID，默认也不会显示，当使用-g 参数时可打开。

（5）USER：命令的执行 UID 或系统中登录的用户名。默认显示为用户名。当使用-l 参数时，可显示 UID。

（6）FD：文件的 File Dsscriptor number。

（7）TYPE。

- IPv4：IPv4 的包。
- IPv6：使用 IPv6 格式的包。
- DIR：目录。
- LINK：链接文件。

（8）DEVICE：使用 character special、block special 表示的设备号。

（9）SIZE：文件的大小；如果不能用大小表示，会留空。使用-s 参数控制。

（10）NODE：本地文件的 node 码，或者协议，如 TCP 等。

（11）NAME：挂载点和文件的全路径（链接会被解析为实际路径），或者链接双方的地址和端口、状态等。

应用实例如下。

（1）查看网卡 22 端口连接，如图 5-6 所示。

```
# lsof -i:22
```

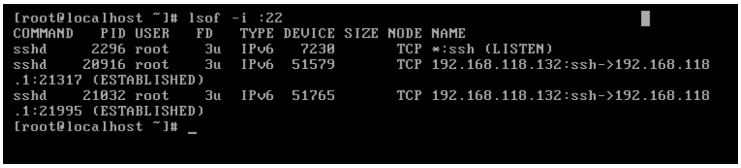


图 5-6　查看网卡 22 端口连接

网卡 22 端口是 ssh 连接端口。

（2）查看目前的 FTP 连接情况。

```
#lsof -i tcp@ohaha.ks.cn:ftp –r
```

（3）查看进程号为 12 的进程打开了哪些文件。

```
#lsof -p 12
```

5.34 ethtool：查询及设置网卡参数

作用：ethtool 是一个专业修改和查看网卡的传输速率、半双工、全双工、网卡连接状态、自适应工作方式等的命令，功能非常强大。

用法：ethtool[选项]设备

主要选项如下。

- -h：显示命令帮助。
- -i：查询网卡的相关信息，如驱动的名称、版本等。
- -d：查询网卡注册性信息。
- -r：重置网卡到自适应模式。
- -S：查询网卡收发包统计。
- -a：查看网卡中接收模块 RX、发送模块 TX 和 Autonegotiate 模块的状态；启动 on 或停用 off。
- -A：修改网卡中接收模块 RX、发送模块 TX 和 Autonegotiate 模块的状态；启动 on 或停用 off。
- -e：显示 EEPROMdump 信息，部分网卡驱动不支持该选项。
- -E：修改网卡 EEPROMbyte。
- -k：显示网卡 Offload 参数的状态——on 或 off，包括 rx-checksumming、tx-checksumming 等。
- -K：修改网卡 Offload 参数的状态。
- -p：用于区别不同 ethX 对应网卡的物理位置，常用的方法是使网卡 port 上的 led 不断地闪；N 指示了网卡闪的持续时间，以秒为单位。
- -r：如果 auto-negotiation 模块的状态为 on，则 restartsauto-negotiation。
- -S：显示 NIC-anddriver-specific 的统计参数，如网卡接收/发送的字节数、接收/发送的广播包个数等。
- -t：让网卡执行自我检测，有两种模式——offline 或 online。
- -s：修改网卡的部分配置，包括网卡速度、单工/全双工模式、MAC 地址等。
- -s：[speed10|100|1000]\：设置网卡速率为 10/100/1000Mbit/s。
- [duplex half|full]\：设置网卡半/全双工。
- [autoneg on|off]\：设置网卡是否自协商。
- [port tp|aui|bnc|mii]\：设置网卡类型。

应用实例如下。

（1）查看网卡的工作情况。

```
#ethtool eth0
Settings for eth0:
        Supported ports: [MII ]
        Supported link modes:   10baseT/Half 10baseT/Full
                               100baseT/Half 100baseT/Full
                                   1000baseT/Full
        Supports auto-negotiation: Yes
        Advertised link modes:  10baseT/Half 10baseT/Full
                               100baseT/Half 100baseT/Full
                               1000baseT/Full
        Advertised auto-negotiation: Yes
        Speed: 100Mb/s          //传输速率为100Mbit/s
        Duplex: Full            //工作方式为全双工
        Port: MII
        PHYAD: 1
        Transceiver: external
        Auto-negotiation: on    //自适应打开
        Supports Wake-on: g
        Wake-on: d
        Link detected: yes      //网线连接正常，即有网线连接。如果为no，表示此
网卡没有激活、启用或网线没有连接
```

（2）设置 eth1 网卡，自适应关闭，传输速率为 1000Mbit/s，工作模式为半双工。

```
# ethtool -s eth1 autoneg off speed 1000 duplex half
```

5.35 netconf：设置各项网络功能

作用：netconf 是老版本的 Red Hat Linux 发行版专门用来调整 Linux 各项设置的程序，目前已不使用。

用法：netconf[选项]

主要选项如下。

- --bootproto=(dhcp|bootp|none)：使用 dhcp|bootp。
- --gateway=STRING：设置网关。
- --ip=STRING：设置 IP 地址。
- --nameserver=STRING：设置 DNS 服务器。
- --netmask=STRING：设置子网掩码。

- --hostname=STRING：设置主机名。
- --domain=STRING：设置域名。
- -d, --device=STRING：网卡设备名（eth0/eth1 之类的网卡设备名）。
- --nodns：不进行 DNS 查询。

说明

netconf 通过对话的方式配置一个永久 IP 地址、网关及 DNS 服务器信息。通过 netconfig 命令设置的 IP 信息会直接写到网卡配置文件（/etc/sysconfig/network-scripts/ifcfg-eth0）和 DNS 客户端配置文件（/etc/resolv.conf）中，但不会立即生效。设置完成后还需重启 network 服务或重启计算机系统。netconf 在最新版 Red Hat Linux 中默认情况下没有安装，用户要手工安装后使用，步骤如下：

```
#mount /dev/cdrom /mnt
#cd /mnt/Servers
#rpm -ivh netconfig-0.8.24-1.2.2.1.i386.rpm
```

应用实例如下。

配置 eth0 网卡的 IP 地址为 192.168.100.1，子网掩码为 255.255.255.0，网关为 192.16.100.254，DNS 为 192.16.100.1。

```
# netconfig --device eth0   //设置 eth0 网卡 IP 地址，此时的--device 可以省略
```

弹出如图 5-7 所示的界面，选择“Yes”，进入配置界面。

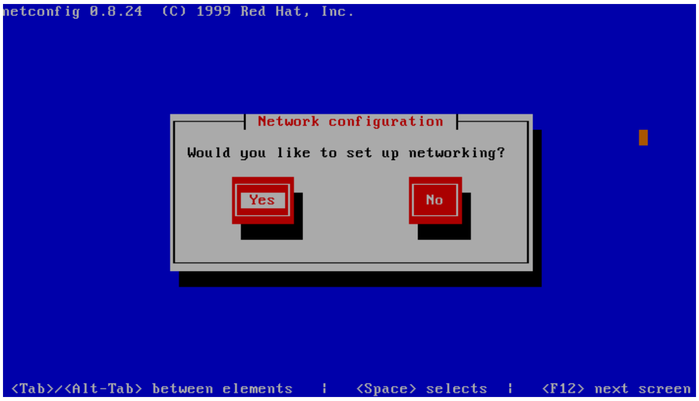


图 5-7　更改网卡配置文件

弹出如图 5-8 所示的界面，若在单选框中用空格选定时，表示 eth0 采用 DHCP 方式动态获取 IP 地址。若不选定单选框，则在对应的文本框中输入 IP 地址、子网掩码、网关、DNS 服务器 IP 地址信息。输入完成后，选择“OK”确定。

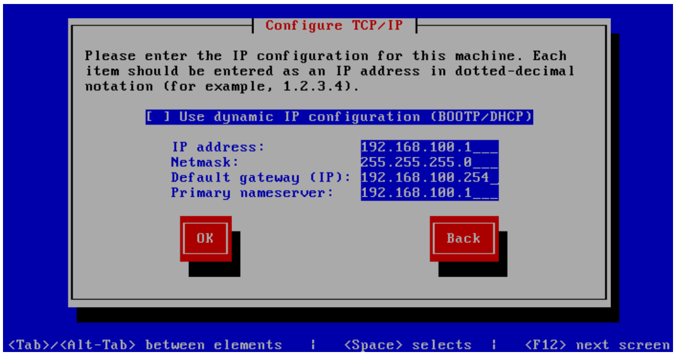


图 5-8 配置网卡静态 IP 地址

5.36 tc：显示和维护流量控制设置

作用：Linux 操作系统中的流量控制器 TC（Traffic Control）用于 Linux 内核的流量控制，主要通过在输出端口处建立一个队列来实现流量控制。

用法：tc [选项]主体{命令| help }

主要选项如下。

- -s：输出更为详尽的信息。
- -d：输出详细信息。
- -batch file：使用批处理命令。

主体如下。

流量的处理由 3 种对象控制，它们是 qdisc（排队规则）、class（类别）和 filter（过滤器）。

- qdisc（排队规则）：qdisc（排队规则）是 queueing discipline 的简写，它是理解流量控制（Traffic Control）的基础。无论何时，内核如果需要通过某个网络接口发送数据包，它都需要按照为这个接口配置的 qdisc（排队规则）把数据包加入队列。然后，内核会尽可能多地从 qdisc 里面取出数据包，把它们交给网络适配器驱动模块。

最简单的 qdisc 是 pfifo，它不对进入的数据包做任何处理，数据包采用先入先出的方式通过队列。不过，它会保存网络接口一时无法处理的数据包。

- class（类别）：某些 qdisc（排队规则）可以包含一些类别，不同的类别中可以包含更深入的 qdisc（排队规则），通过这些细分的 qdisc 还可以为进入队列的数据包排队。通过设置各种类别数据包的离队次序，qdisc 可以设置网络数据流量的优先级。
- filter（过滤器）：filter（过滤器）用于为数据包分类，决定它们按照何种 qdisc 进入队列。无论何时数据包进入一个划分子类的类别中，都需要进行分类。分类的方法可以有多种，使用 fileter（过滤器）就是其中之一。使用 filter（过滤器）分类时，内核会调用附属于这个类（class）的所有过滤器，直到返回一个判决。如果没有判决返回，就做进一步的处理，而处理方式和 qdisc 有关。

tc 可以使用以下命令。

- add：在一个节点里加入一个 qdisc、类或者过滤器。添加时，需要传递一个祖先作为参数，传递参数时既可以使用 ID，也可以直接传递设备的根。如果要建立一个 qdisc 或者过滤器，可以使用句柄（handle）来命名；如果要建立一个类，可以使用类识别符（classid）来命名。
- remove：删除由某个句柄（handle）指定的 qdisc，根 qdisc（root）也可以删除。被删除的 qdisc 上的所有子类及附属于各个类的过滤器都会被自动删除。
- change：以替代的方式修改某些条目。除了句柄（handle）和祖先不能修改以外，change 命令的语法和 add 命令相同。换句话说，change 命令不能移动节点的位置。
- replace：对一个现有节点进行近于原子操作的删除/添加。如果节点不存在，这个命令就会建立节点。
- link：只适用于 dqisc，替代一个现有的节点。

应用实例如下。

对服务器 1935 端口进行限速，限制上传和下载速率都为 55kbit/s。

```
    # tc qdisc del dev eth0 root
    # tc qdisc add dev eth0 root handle 1:0 cbq bandwidth 100Mbit avpkt 1000
cell 8
    # tc class add dev eth0 parent 1:0 classid 1:1 cbq bandwidth 100Mbit rate
55Kbit weight 55Kbit prio 8 allot 1514 cell 8 maxburst 20 avpkt 1000 bounded
    # tc class add dev eth0 parent 1:1 classid 1:4 cbq bandwidth 100Mbit rate
55Kbit weight 55Kbit prio 5 allot 1514 cell 8 maxburst 20 avpkt 1000
    # tc qdisc add dev eth0 parent 1:4 handle 40: sfq
    # tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 match ip sport
1935 0xffff flowid 1:4
```

5.37 telnet：远程登录

作用：telnet 命令开启终端机阶段作业，并登入远端主机。

用法：telnet [-8acdEfFKLrx] [-b<主机别名>] [-e<脱离字符>] [-k<域名>] [-l<用户名称>] [-n<记录文件>] [-S<服务类型>] [-X<认证形态>] [主机名称或 IP 地址<通信端口>]

主要选项如下。

- -8：允许使用 8 位字符资料，包括输入与输出。
- -a：尝试自动登入远端系统。
- -b<主机别名>：使用别名指定远端主机名称。
- -c：不读取用户专属目录里的.telnetrc 文件。
- -d：启动排错模式。

- -e<脱离字符>：设置脱离字符。
- -E：滤除脱离字符。
- -f：此参数的效果和指定“-F”参数相同。
- -F：使用 Kerberos V5 认证时，加上此参数可把本地主机的认证数据上传到远端主机。
- -k<域名>：使用 Kerberos 认证时，加上此参数让远端主机采用指定的领域名，而非该主机的域名。
- -K：不自动登入远端主机。
- -l<用户名称>：指定要登入远端主机的用户名称。
- -L：允许输出 8 位字符资料。
- -n<记录文件>：指定文件记录相关信息。
- -r：使用类似 rlogin 指令的用户界面。
- -S<服务类型>：设置 telnet 连线所需的 IP TOS 信息。
- -x：假设主机有支持数据加密的功能，就使用它。
- -X<认证形态>：关闭指定的认证形态。

应用实例如下。

使用 telnet 命令登录一台 Windows XP 主机。

首先检查该主机是否安装 Telnet 服务。

```
#rpm -qa |grep  telnet
```

若没有，请安装 Telnet 服务。

```
#rpm -ivh /media/cdrom/rhel5/telnet*.rpm    #以 telnet 开头的包全部安装
```

开启服务。

```
#vi/etc/xinetd.d/telnet 改写“disable=no”；
```

激活 Telnet 服务。

```
#service xinetd restart
```

创建一个用于 telnet 的用户#useradd cjh。

```
#passwd cjh
#******             #根据提示创建用户密码
```

给 Windows XP 创建一个账户（要设置密码），用于 Linux“telnet”Windows XP。

在运行界面中输入“services.msc”打开服务，将 Telnet 服务启动。

使用 telnet 命令登录。

```
# telnet 192.168.118.1
```

连接后输入用户名和密码，成功登录界面如图 5-9 所示。

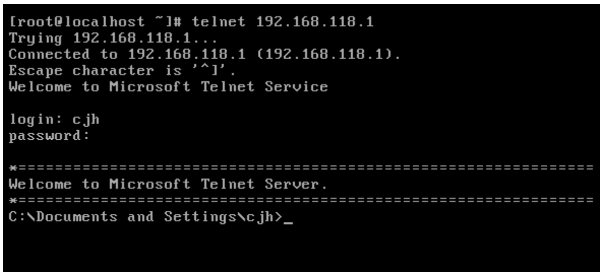


图 5-9　使用 telnet 登录一台 Windows XP 主机

> 注意
>
> 出于安全考虑，现在 telnet 命令已经很少使用了。

5.38　rlogin：远程登录命令

作用：rlogin（Remote Login in UNIX Systems）是一个 UNIX/Linux 命令，它允许授权用户进入网络中的其他 UNIX 机器，并且就像用户在现场操作一样。一旦进入主机，用户就可以操作主机允许的任何事情，比如读文件、编辑文件和删除文件等。

用法：rlogin [-8EL] [-e <脱离字符>] [-l <用户名称>] [主机名称或 IP 地址]

主要选项如下。

- -E：忽略 escape 字符。
- -8：只识别 8 位字的字符。
- -L：允许 rlogin 会话运行在 litout 模式。
- -ec：设置 escape 字符为 c。
- -c：断开连接前要求确认。
- -a：强制要求远程主机在发送完一个空的本地用户名之后请求一个密码。
- -f：向远程主机发送一个本地认证。
- -F：向远程主机发送一个可转寄的本地认证。
- -7：强制执行 7 位的传输。
- -d：打开用于远程主机通信的 TCP 套接口的调试。
- -k：要求包含远程主机的 tickets。
- -x：启动数据传输的 DES 加密。
- -4：只使用 Kerberos 的版本 4 的认证。
- -e<字符>：设置退出字符。
- -l<用户>：指定登录的用户。
- -t<终端类型>：设置终端类型。

应用实例如下。

（1）使用指定用户名登录远程主机，如图 5-10 所示。

```
# rlogin 192.168.118.132 -l cjh
```

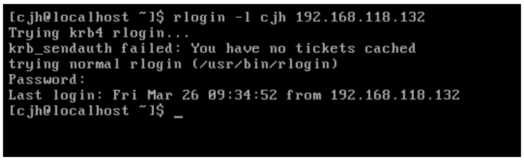


图 5-10　使用指定用户名登录远程主机

为什么会出现 Kerberos 认证错误，其实是因为 RHEL5 里面有两个 rlogin 程序，一个是基于 Kerberos 认证的 rlogin，一个是普通的 rlogin。系统会优先使用/usr/kerberos/bin 下面的 rlogin。由于没有配置基于 Kerberos 的 rlogin，所以会出现 543 端口连接不上、认证失败的提示。/usr/kerberos/bin/rlogin 一旦发现认证失败了，就会调用/usr/bin/rlogin 用普通的 rlogin 尝试登录服务器，因此第二次就成功了。

（2）如何允许 root 通过 rlogin 登录？只要把 rlogin 添加进/etc/securetty 即可。

说明

每一个远程机器都有一个文件（/etc/hosts.equiv），包括了一个信任主机名集共享用户名的列表。本地用户名和远程用户名相同的用户，可以在/etc/hosts.equiv 文件中列出的任何机器上登录到远程主机，而不需要密码口令。个人用户可以在主目录下设置相似的个人文件（通常叫.rhosts）。此文件中的每一行都包含了两个名字——主机名和用户名，二者用空格分开。.rhosts 文件中的每一行包括允许一个登录到主机名的用户名称，该用户无须密码就可以登录远程主机。如果在远程主机的/etc/hosts.equiv 文件中找不到本地主机名，并且在远程用户的.rhosts 文件中找不到本地用户名和主机名，远程主机就会提示密码。列在/etc/hosts.equiv 和.rhosts 文件中的主机名必须是列在主机数据库中的正式主机名，昵称均不许使用。为安全起见，.rhosts 文件必须归远程用户或根所有。远程终端类型和本地终端类型（在 TERM 变量环境中给定）相同。如果服务器支持，终端或窗口尺寸会被复制到远程系统中，同时大小的变化也能反映出来。所有的回声现象都发生在远程站点，以至于远程登录都是透明的（除了延迟情况）。流控制借助<CTRL-S>和<CTRL-Q>实现，并且输入/输出中断也能得到很好的处理。

5.39 rsh：远程登录的 Shell

作用：rsh（remote shell）提供用户环境，也就是 Shell，以便指令能够在指定的远程主机上执行。

用法：rsh [-dn] [-l <用户名称>] [主机名称或 IP 地址] [执行指令]

主要选项如下。

- -d：使用 Socket 层级的排错功能。
- -l<用户名称>：指定要登录远程主机的用户名称。
- -n：重定向来自特殊设备/dev/null 的输入。

应用实例如下。

远程登录一台主机。

```
$ rsh 192.168.118.132
Trying krb4 rlogin...
krb_sendauth failed: You have no tickets cached
trying normal rlogin (/usr/bin/rlogin)
Password:
Last login: Fri Mar 26 09:43:17 from 192.168.118.132
[cjh@localhost ~]$
```

5.40 usernetctl：让普通用户控制网络接口

作用：特权用户用 usernetctl 使其他用户可以用 ifup 和 ifdown 等命令。

用法：usernetctl <interface-config> <up|down|report>

主要选项如下。

- interface-name：被操纵的网络接口的名称，例如可以是 ppp0 或 eth0。
- up：激活网络接口。
- down：禁用网络接口。
- report：报告网络接口状态。

说明

普通用户一般不需要去开/关网络接口，特殊情况下，需要普通用户拥有这一权力，例如开启 ppp 或者 vpn 连接。例如我们需要用户能开启/关闭一个网络接口 eth0，此时可以使用这个命令。

应用实例如下。

让普通用户可以启动 eth0 网卡。

```
# usernetctl eth0 up
```

5.41 nmcli：NetworkManager 命令行网络接口配置工具

作用：使用 nmcli 命令用户可以查询网络连接的状态，也可以用来管理。

用法：nmcli [OPTIONS] OBJECT { COMMAND | help }

（1）OPTIONS 是一些修改 IP 行为者改变其输出的选项。所有的选项都以“-”字符开头，分为长、短两种形式。选项如表 5-3 所示。

表 5-3　OPTIONS 选项

选　项	含　义
-t[erse]	简洁输出模式。这个模式适合于计算机（脚本）的处理
-p[retty]	详细输出模式，这个模式适合于用户阅读
-m[mode] tabular | multiline	在表格和多输出之间切换。默认值是表格
-f[ields] <field1,field2,...> | all | common	该选项用于指定哪些字段（列名）可以被打印出来
-e[scape] yes | no	“:” .和'\'字符简洁的表格模式
-v	打印 IP 的版本并退出
-help	显示帮助信息
-a[sk]	要求缺少的参数
w[ait] <seconds>	设置超时等待操作完成

（2）OBJECT 是管理者获取信息的对象。对象包括如表 5-4 所示的选项。

表 5-4　OBJECT 选项

选　项	含　义
n[etworking]	使用此对象查询和修改 Network Manager 的状态。 子命令包括 status | sleep | wakeup | wifi | wwan。 status：显示 Network Manager 的整体状态。 sleep：把 Network Manager 设置为睡眠模式。此时 Network Manager 管理的所有接口都被停用。 wakeup：从睡眠模式下唤醒。 wifi：查询或设置 Network Manager 的无线网络设备的状态。 wwan：查询或设置 Network Manager 的 WWAN 网络设备的状态
c[onnection]	获取有关 NetworkManager 的连接信息
d[evice]	获取有关设备的信息
r[adio]	NetworkManager 的无线交换机
g[eneral]	NetworkManager 的一般状况和运作

（3）COMMAND 子命令选项如下。

```
list [id <id> | uuid <id> | system | user]
```

配置表的连接。如果没有参数，配置为从系统和用户设置的服务中列出的连接。为了获得连接信息，标识与连接的名称（UUID）必须被指定。

```
status
```

打印活动连接的状态。

```
up id <id> | uuid <id> [iface <iface>] [ap <hwaddr>] [--nowait] [--timeout <timeout>]
```

激活的连接。连接识别用的 ID 或 UUID。

```
down id <id> | uuid <id>
```

停用的连接。连接识别用的 ID 或 UUID。

```
disconnect iface <iface> [--nowait] [--timeout <timeout>]
```

断开一个设备，防止设备自动启动。

```
wifi [list [iface <iface>] [hwaddr <hwaddr>]]
```

列出可用的 WiFi 接入点。

应用实例如下。

（1）查看网络接口状态。

```
# nmcli -p g
=====================================================================
                         网络管理器状态
=====================================================================
状态                    CONNECTIVITY  WIFI-HW  WIFI    WWAN-HW  WWAN
---------------------------------------------------------------------
connected (local only)  full          已启用    已启用   已启用    已禁用
```

> **说明**
>
> 这里使用-p 选项是为了获取详细的输出情况。另外，g 是 general 的缩写。这个命令支持缩写输入。

（2）查看有关权限的一些信息。

```
# nmcli g permissions
PERMISSION                                                VALUE
org.freedesktop.NetworkManager.enable-disable-network     是
org.freedesktop.NetworkManager.enable-disable-wifi        是
org.freedesktop.NetworkManager.enable-disable-wwan        是
```

```
org.freedesktop.NetworkManager.enable-disable-wimax             是
org.freedesktop.NetworkManager.sleep-wake                       是
org.freedesktop.NetworkManager.network-control                  是
org.freedesktop.NetworkManager.wifi.share.protected             是
org.freedesktop.NetworkManager.wifi.share.open                  是
org.freedesktop.NetworkManager.settings.modify.system           是
org.freedesktop.NetworkManager.settings.modify.own              是
org.freedesktop.NetworkManager.settings.modify.hostname         是
```

（3）查看日志信息。

```
# nmcli g logging
LEVEL  DOMAINS
INFO
PLATFORM,RFKILL,ETHER,WIFI,BT,MB,DHCP4,DHCP6,PPP,IP4,IP6,AUTOIP4,DNS,VP
N,SHARING,SUPPLICANT,AGENTS,SETTINGS,SUSPEND,CORE,DEVICE,OLPC,WIMAX,INFINIB
AND,FIREWALL,ADSL,BOND,VLAN,BRIDGE,TEAM,CONCHECK,DCB
```

（4）查看网络是否启用。

```
# nmcli networking
```

已启用。

（5）获取所有可用的网络配置的列表。

```
# nmcli connection
名称                              UUID                          类型    设备
virbr0          cb1d36f2-be28-4f11-96a6-fed67a1e3c82       bridge virbr0
Wired connection 1 d4e44a18-4f63-4a5f-b8a3-e7d9470acf78 802-3-ethernet enp0s3
enp0s3         df3ebf2a-2a6c-4618-b166-c692cb3abd32 802-3-ethernet --
```

（6）查看主机名称。

```
# nmcli general hostname
localhost.localdomain
```

（7）修改主机名称为 cjh。

```
#  nmcli general hostname cjh
```

（8）查看网络设备列表。

```
# nmcli device show
GENERAL.设备:           wlp0s29f7u4
GENERAL.类型:           wifi
GENERAL.硬盘:           14:E6:E4:29:6C:16
GENERAL.MTU:            1500
```

```
GENERAL.状态:            100 (连接的)
GENERAL.CONNECTION:     自动 TP-LINK_BB006E
GENERAL.CON-PATH:       /org/freedesktop/NetworkManager/ActiveConnection/0
IP4.地址[1]:             ip = 192.168.1.106/24, gw = 192.168.1.1
IP4.DNS[1]:             101.226.4.6
IP4.DNS[2]:             8.8.8.8
IP6.地址[1]:             ip = fe80::16e6:e4ff:fe29:6c16/64, gw = ::

GENERAL.设备:            virbr0
GENERAL.类型:            bridge
GENERAL.硬盘:            E2:FA:0A:82:15:AD
GENERAL.MTU:            1500
GENERAL.状态:            70 (连接中(获得 IP 配置))
GENERAL.CONNECTION:     virbr0
GENERAL.CON-PATH:       /org/freedesktop/NetworkManager/ActiveConnection/1
GENERAL.设备:            enp1s0
GENERAL.类型:            ethernet
GENERAL.硬盘:            00:19:66:DF:32:EF
GENERAL.MTU:            1500
GENERAL.状态:            20 (不可用)
GENERAL.CONNECTION:     --
GENERAL.CON-PATH:       --
WIRED-PROPERTIES.容器:关

GENERAL.设备:            lo
GENERAL.类型:            loopback
GENERAL.硬盘:            00:00:00:00:00:00
GENERAL.MTU:            65536
GENERAL.状态:            10 (未管理)
GENERAL.CONNECTION:     --
GENERAL.CON-PATH:       --
IP4.地址[1]:             ip = 127.0.0.1/8, gw = 0.0.0.0
IP6.地址[1]:             ip = ::1/128, gw = ::
```

（9）查看一个指定的网络设置，这里指定为无线网卡。

```
#  nmcli device show  wlp0s29f7u4
GENERAL.设备:            wlp0s29f7u4
GENERAL.类型:            wifi
GENERAL.硬盘:            14:E6:E4:29:6C:16
```

```
GENERAL.MTU:             1500
GENERAL.状态:             100 (连接的)
GENERAL.CONNECTION:      自动 TP-LINK_BB006E
GENERAL.CON-PATH:        /org/freedesktop/NetworkManager/ActiveConnection/0
IP4.地址[1]:              ip = 192.168.1.106/24, gw = 192.168.1.1
IP4.DNS[1]:              101.226.4.6
IP4.DNS[2]:              8.8.8.8
IP6.地址[1]:              ip = fe80::16e6:e4ff:fe29:6c16/64, gw = ::
```

（10）显示无线热点。

```
# nmcli d wifi
*  SSID                     型号   CHAN  频率      信号   BARS   安全性
   hyp_home                 Infra  13    54 MB/s  64    ▂▄▆_   WPA2
   superzhang               Infra  1     54 MB/s  57    ▂▄▆_   WPA1 WPA2
   TP-LINK_BA37             Infra  1     54 MB/s  40    ▂▄__   WPA1 WPA2
   gehua03111406180949960   Infra  1     54 MB/s  45    ▂▄__   WPA2
   feng                     Infra  6     54 MB/s  45    ▂▄__   WPA1 WPA2
   2-1202                   Infra  6     54 MB/s  44    ▂▄__   WPA1 WPA2
```

（11）查询 DNS 地址。

```
# nmcli connection show ens3 | grep dns
ipv4.dns:                          192.168.122.1
ipv4.dns-search:
ipv4.ignore-auto-dns:                no
ipv6.dns:
ipv6.dns-search:
ipv6.ignore-auto-dns:                no
```

（12）删除一个网络连接。

```
# nmcli connection delete ens3
```

实际上是删除/etc/sysconfig/network-scripts/ifcfg-ens3 这个文件。

5.42 nmtui：基于 Curses 的用户界面 nmtui

作用：这个版本的 NetworkManager 为用户带来了大量新增特性和改进，包括新增基于 Curses 的用户界面 nmtui，对 nmcli 加入了交互式编辑支持、单命令编辑、详细的帮助、Tab 补全及增强的 bash 补全等功能。

用法：sudo nmtui

该命令无参数。操作界面如图 5-11 所示。

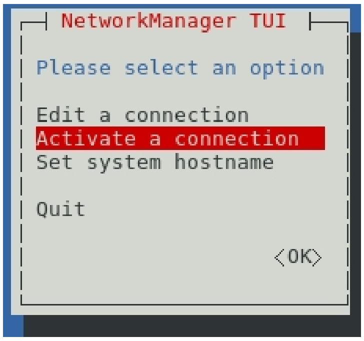


图 5-11　nmtui 工作界面

nmtui 的主界面包括 3 个选项：编辑一个连接、激活一个连接、设置主机名称。

如图 5-12 所示是编辑一个连接的界面。

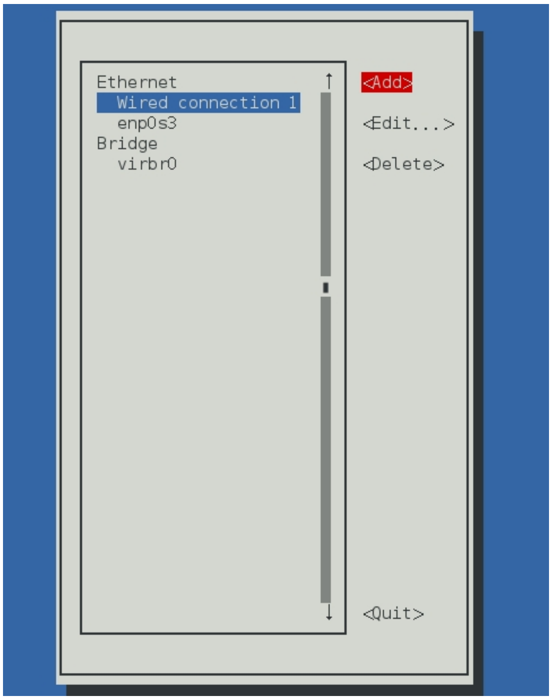


图 5-12　编辑一个连接的界面

图 5-12 所示的界面中有 3 个按钮，分别代表添加一个连接（Add 按钮，如图 5-13 所示）、编辑一个连接（Edit 按钮，如图 5-14 所示）、删除一个连接（Delete 按钮，如图 5-15 所示）。

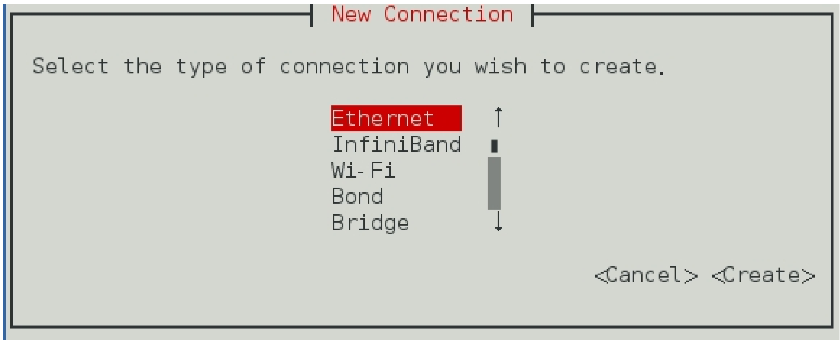


图 5-13　添加一个连接

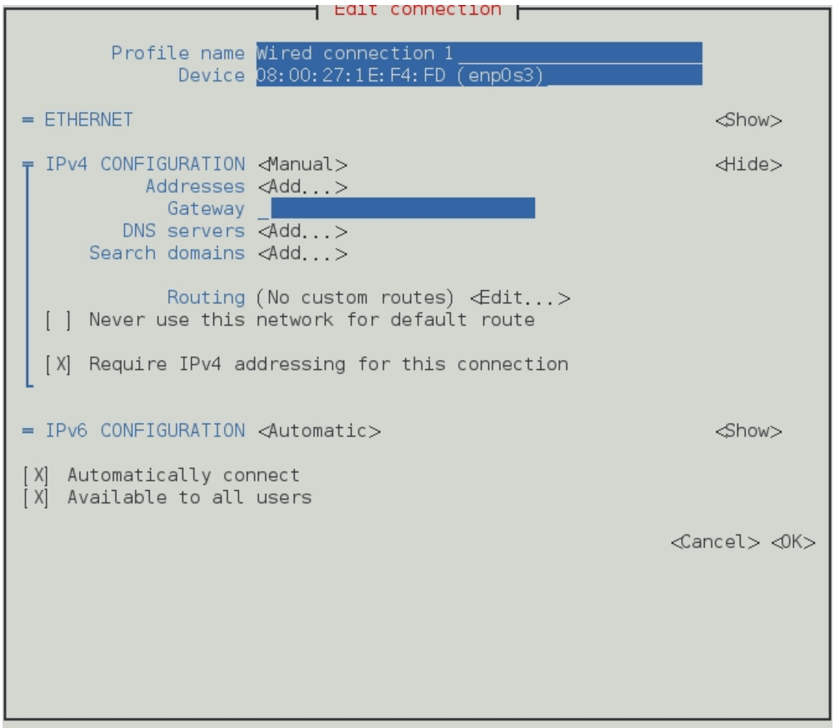


图 5-14　编辑一个连接

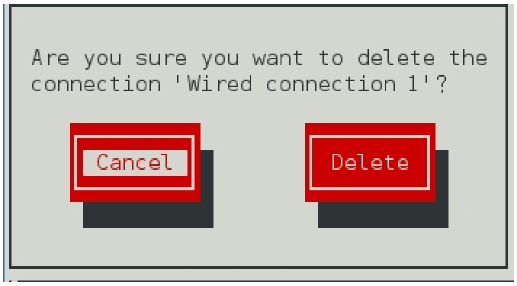


图 5-15　删除一个连接

5.43 nc：netcat，Linux 下用于调试和检查网络的工具包

作用：netcat 是网络工具中的“瑞士军刀”，它能通过 TCP 和 UDP 在网络中读/写数据。通过与其他工具结合和重定向，用户可以在脚本中以多种方式使用它。netcat 所做的就是在两台计算机之间建立链接并返回两个数据流，用户能建立一个服务器，用于传输文件，与朋友聊天，传输流媒体，或者作为其他协议的独立客户端。

用法：nc [-hlnruz] [-g<网关...>] [-G<指向器数目>] [-i<延迟秒数>] [-o<输出文件>] [-p<通信端口>] [-s<来源地址>] [-v...] [-w<超时秒数>] [主机名称] [通信端口...]

主要选项如下。

- -g<网关>：设置路由器跃程通信网关，最多可设置 8 个。
- -G<指向器数目>：设置来源路由指向器，其数值为 4 的倍数。
- -h：在线帮助。
- -i<延迟秒数>：设置时间间隔，以便传送信息及扫描通信端口。
- -l：使用监听模式，管控传入的资料。
- -n：直接使用 IP 地址，而不通过域名服务器。
- -o<输出文件>：指定文件名称，把往来传输的数据以十六进制字码倾倒成该文件保存。
- -p<通信端口>：设置本地主机使用的通信端口。
- -r：乱数指定本地主机与远端主机的通信端口。
- -s<来源地址>：设置本地主机送出数据包的 IP 地址。
- -u：使用 UDP 传输协议。
- -v：显示指令执行过程。
- -w<超时秒数>：设置等待连线的时间。
- -z：使用 0 输入/输出模式，只在扫描通信端口时使用。

应用实例如下。

（1）端口扫描。

```
# nc -v -w 2 192.168.1.34 -z 21-24
nc: connect to 192.168.2.34 port 21 (tcp) failed: Connection refused
Connection to 192.168.2.34 22 port [tcp/ssh] succeeded!
nc: connect to 192.168.2.34 port 23 (tcp) failed: Connection refused
nc: connect to 192.168.2.34 port 24 (tcp) failed: Connection refused
```

（2）传文件。

```
# nc -l 1234 > test.txt#192.168.1.10
# nc 192.168.2.34 < test.txt #192.168.1.11
```

（3）建立从本地 1234 端口到 host.example.com 的 80 端口的连接，5 秒超时。

```
# nc -p 1234 -w 5 blog.51yip.com 80
```

5.44 lnstat：显示 Linux 系统的网络状态

作用：lnstat 命令用来显示 Linux 系统的网络状态，用于替代旧的 rtstat 程序。

用法：lnstat [选项]

主要选项如下。

- -h：显示帮助信息。
- -V：显示指令版本信息。
- -c：指定显示网络状态的次数，每隔一定时间显示一次网络状态。
- -d：显示可用的文件或关键字。
- -i：指定两次显示网络状态的间隔秒数。
- -k：只显示给定的关键字。
- -s：是否显示标题头。
- -w：指定每个字段所占的宽度。

应用实例如下。

显示网络状态的来源文件列表。

```
# lnstat -d
```

5.45 ss：显示网络状态信息

作用：ss 是类似 netstat 的工具，能显示网络状态信息，包括 TCP、UDP 连接和端口。

用法：ss[选项] [FILTER]

主要选项如下。

- -a：显示所有网络连接。
- -l：显示 LISTEN 状态的连接（连接打开）。
- -m：显示内存信息（用于 tcp_diag）。
- -o：显示 TCP 定时器 x。
- -p：显示进程信息。
- -s：连接统计。
- -d：只显示 DCCP 信息（等同于-A dccp）。
- -u：只显示 UDP 信息（等同于-A udp）。
- -w：只显示 RAW 信息（等同于-A raw）。
- -t：只显示 TCP 信息（等同于-A tcp）。
- -x：只显示 UNIX 通信信息（等同于-A unix）。
- -4：只显示 IPv4 信息。
- -6：只显示 IPv6 信息。

- --help：显示帮助信息。
- --version：显示版本信息。

应用实例如下。

（1）显示 ICP 连接。

```
# ss -t -a
```

（2）显示 Sockets 摘要。

```
# ss –s
```

（3）查看进程使用的 Socket。

```
# ss -pl
```

（4）显示所有的 UDP Sockets。

```
# ss -u -a
```

5.46 rexec：在指定的远程 Linux 系统主机上执行命令

作用：rexec 命令用于在指定的远程 Linux 系统主机上执行命令，向远程 rexec 服务器发出执行命令的请求。rexec 命令通过检查$HOME/.netrc 文件（包含远程主机上使用的用户名和密码）来提供自动登录的功能。如果没有发现系统在安全方式下操作，则 rexec 命令提示输入一个远程主机的有效用户名和密码。在这两种情况下，rexec 均导致远程系统上的 rexecd 使用默认的 compat 用户登录认证方法。rexecd 不会为了备用的认证方法去查找/etc/security/user 文件。也可以指定-n 标志到 rexec 命令行上来重设自动登录功能。

用法：rexec [选项] [主机名称]

主要选项如下。

- -a：表示远程命令的标准错误与标准输出相同，不支持发送任意信号到远程进程。
- -l<用户名>：指定连接远程 rexec 服务器的用户名。
- -p<密码>：指定连接远程 rexec 服务器的密码。
- -n：明确地提示输入用户名和密码。
- 主机名称：指定远程主机（IP 地址或主机名）。

应用实例如下。

（1）在一个远程主机上执行 date 命令，输入如下命令：

```
# rexec host1 date
```

date 命令的输出现在显示在本地系统上。在本示例中，在本地主机上的$HOME/.netrc 文件包含远程主机上有效的用户名和密码。如果没有远程主机的$HOME/.netrc 文件中的有效项，将提示输入登录标识和密码。输入所要求的登录信息后，date 命令的输出显示在本地系统上。

（2）重设自动登录功能并执行远程主机上的 date 命令，输入如下命令：

```
#rexec -nhost1
```

date 出现提示时输入用户名和密码，date 命令的输出现在显示在本地系统上。

（3）列出远程主机上另一个用户的目录，输入如下命令：

```
# rexec host1 ls -l /home/karen
```

在远程主机 host1 上的 karen 用户的目录列表显示在本地系统上。如果没有远程主机的 $HOME/.netrc 文件中的有效项，将提示输入登录标识和密码。输入要求的登录信息后，在远程主机 host1 上的 karen 用户的目录列表显示在本地系统上。

06 Linux 用户管理命令

Red Hat Linux 提供了一个图形配置工具 redhat-config-user，但是 Linux 管理员还是应当使用专用命令来完成 Linux 用户和用户组的管理工作。Linux 用户与用户组管理命令与 redhat-config-user 相比具有许多优点。

- 命令更加灵活，使用命令通常比使用 redhat-config-user 有更多的选项。
- 命令运行更快，不必等待 Linux 处理图形。
- redhat-config-user 工具不能显示所有的错误信息，而命令提供的错误信息可以在控制台看到。
- redhat-config-user 工具是另一层软件，也是另一个出错源。

6.1 useradd：建立用户

作用：建立用户账号和创建用户的起始目录。

用法：useradd [选项] 登录

useradd -D

useradd -D [选项]

主要选项如下。

- -b,--base-dir BASE_DIR：新账户的主目录的基目录。
- -c,--comment COMMENT：新账户的 GECOS 字段。
- -d,--home-dir HOME_DIR：新账户的主目录。
- -D,--defaults：显示或更改默认的 useradd 配置。
- -e,--expiredate EXPIRE_DATE：新账户的过期日期。
- -f,--inactive INACTIVE：新账户的密码不活动期。
- -g,--gid GROUP：新账户主组的名称或 ID。
- -G,--groups GROUPS：新账户的附加组列表。
- -h,--help：显示此帮助信息并退出。
- -k,--skel SKEL_DIR：使用此目录作为骨架目录。
- -K,--key KEY=VALUE：不使用/etc/login.defs 中的默认值。
- -l,--no-log-init：不要将此用户添加到最近登录和登录失败的数据库。

- -m,--create-home：创建用户的主目录。
- -M,--no-create-home：不创建用户的主目录。
- -N,--no-user-group：不创建同名的组。
- -o,--non-unique：允许使用重复的 UID 创建用户。
- -p,--password PASSWORD：加密后的新账户密码。
- -r,--system：创建一个系统账户。
- -R,--root CHROOT_DIR：chroot 到的目录。
- -s,--shell SHELL：新账户的登录 Shell。
- -u,--uid UID：新账户的用户 ID。
- -U,--user-group：创建与用户同名的组。
- -Z,--selinux-user SEUSER：为 SELinux 用户映射使用指定的 SEUSER。

说明

useradd 可用来建立用户账号，它和 adduser 命令是完全相同的，adduser 是 useradd 的符号链接。增加用户账号就是在/etc/passwd 文件中为新用户增加一条记录，同时更新其他系统文件，如/etc/shadow、/etc/group 等。账号建好之后，通常用 passwd 设定账号的密码。

应用实例如下。

（1）创建普通用户。

```
#useradd –d /usr/cjh -m cjh
```

上面的命令创建一个用户 cjh，其登录名为 cjh，并产生一个主目录/usr/cjh（/usr 为默认的用户主目录所在的父目录）。

（2）新建一个特定用户 gem，该用户的登录 Shell 是/bin/sh，它属于 group 用户组，同时又属于 adm 和 root 用户组，其中 group 用户组是其主组。

```
# useradd -s /bin/sh -g group –G adm,root gem
```

（3）查看 useradd 命令的默认设置。

```
# useradd –D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

默认设置保存在文件/etc/default/useradd 中。

6.2 userdel：删除用户

作用：删除一个用户。

用法：userdel [-r]用户名

主要选项如下。

- -h,--help：显示此帮助信息并退出。
- -r,--remove：删除主目录和邮件池。
- -R,--root CHROOT_DIR：chroot 到的目录。
- -Z,--selinux-user：为用户删除所有的 SELinux 用户映射。
- -f：强制删除用户，即使用户当前已登录。

应用实例如下。

删除一个用户。

```
# userdel -r cjh
```

此命令删除用户 cjh 在系统文件中（/etc/passwd、/etc/shadow、/etc/group）的记录，同时删除用户的主目录。

6.3 usermod：修改已有用户的信息

作用：usermod 命令修改已有用户的信息，即更改用户的有关属性，如用户 ID、主目录、用户组、登录 Shell 等。

用法：usermod [选项] 登录

主要选项如下。

- -c,--comment 注释：GECOS 字段的新值。
- -d,--home HOME_DIR：用户的新主目录。
- -e,--expiredate EXPIRE_DATE：设定账户过期的日期为 EXPIRE_DATE。
- -f,--inactive INACTIVE：过期 INACTIVE 天数后，设定密码为失效状态。
- -g,--gid GROUP：强制使用 GROUP 为新主组。
- -G,--groups GROUPS：新的附加组列表 GROUPS。
- -a,--append GROUP：将用户追加至上边-G 中提到的附加组中，并不从其他组中删除此用户。
- -h,--help：显示此帮助信息并退出。
- -l,--login LOGIN：新的登录名称。
- -L,--lock：锁定用户账号。
- -m,--move-home：将家目录内容移至新位置（仅与-d 一起使用）。
- -o,--non-unique：允许使用重复的（非唯一的）UID。

- -p,--password PASSWORD：将加密过的密码（PASSWORD）设为新密码。
- -R,--root CHROOT_DIR：chroot 到的目录。
- -s,--shell SHELL：该用户账号的新登录 Shell。
- -u,--uid UID：用户账号的新 UID。
- -U,--unlock：解锁用户账号。
- -Z,--selinux-user SEUSER：用户账户的新 SELinux 用户映射。

应用实例如下。

将用户 cjh 的登录 Shell 修改为 tcsh，主目录改为/home/cjh，用户组改为 developer。

```
# usermod -s /bin/ tcsh -d /home/cjh –g developer cjh
```

6.4 passwd：设置密码

作用：修改账户的登录密码。

用法：passwd [选项] <账户名称>

主要选项如下。

- -k,--keep-tokens：保持身份验证令牌不过期
- -d,--delete：删除已命名账号的密码（只有根用户才能进行此操作）。
- -l,--lock：锁定指定账户的密码（仅限 root 用户）。
- -u,--unlock：解锁指定账户的密码（仅限 root 用户）。
- -e,--expire：终止指定账户的密码（仅限 root 用户）。
- -f,--force：强制执行操作
- -x,--maximum=DAYS：密码的最长有效时限（只有根用户才能进行此操作）。
- -n,--minimum=DAYS：密码的最短有效时限（只有根用户才能进行此操作）。
- -w,--warning=DAYS：在密码过期前多少天开始提醒用户（只有根用户才能进行此 操作）。
- -i,--inactive=DAYS：当密码过期后经过多少天该账号会被禁用（只有根用户才能进行此操作）。
- -S,--status：报告已命名账号的密码状态（只有根用户才能进行此操作）。
- --stdin：从标准输入读取令牌（只有根用户才能进行此操作）。

应用实例如下。

（1）修改用户密码。

假设当前用户是 cjh，则下面的命令用于修改该用户自己的密码。

```
$ passwd
Old password:******
New password:*******
Re-enter new password:*******
```

如果是超级用户，可以用下列形式指定任何用户的密码。

```
# passwd cjh
New password:*******
Re-enter new password:*******
```

说明

普通用户修改自己的密码时，passwd 命令会先询问原密码，验证后再要求用户输入两遍新密码，如果两次输入的密码一致，则将这个密码指定给用户；而超级用户为用户指定密码时，就不需要知道原密码。为了保证系统安全，用户应该选择比较复杂的密码，例如，最好使用 8 位长的密码，密码中应包含大写、小写字母和数字，并且应该与姓名、生日等不相同。

（2）为用户指定空密码。

```
# passwd -d cjh
```

该命令将用户 cjh 的密码删除，这样用户 cjh 下一次登录时，系统不再询问密码。

（3）临时锁定一个用户登录。

有时需要让一个用户暂时不能登录，可以使用如下命令（只有超级用户才可以使用这个选项）：

```
# passwd -l cjh
```

（4）解锁用户账号。

恢复被锁定用户的登录权限，可以使用如下命令（只有超级用户才可以使用这个选项）：

```
# passwd -u cjh
```

6.5 chage：密码老化

作用：chage 命令起到密码老化的作用。

用法：chage [选项] 登录

主要选项如下。

- -d,--lastday 最近日期：将最近一次密码设置时间设为“最近日期”。
- -E,--expiredate 过期日期：将账户过期时间设为“过期日期”。
- -h,--help：显示此帮助信息并退出。
- -I,--inactive INACITVE：过期 INACTIVE 天数后，设定密码为失效状态。
- -l,--list：显示账户年龄信息。

– -m,--mindays 最小天数：将两次改变密码之间相距的最小天数设为“最小天数”。
– -M,--maxdays 最大天数：将两次改变密码之间相距的最大天数设为“最大天数”。
– -R,--root CHROOT_DIR：chroot 到的目录。
– -W,--warndays 警告天数：将过期警告天数设为“警告天数”。

应用实例如下。

如果系统管理员想让用户 cjh 在首次登录时设置密码，用户的密码可以被设置为立即过期，从而强制用户 cjh 在首次登录后立即改变它。

```
# chaged 0 cjh
```

6.6 groupadd：添加组

作用：添加新组到系统中。

使用权限：超级用户。

用法：groupadd [选项] group

主要选项如下。

– -f,--force：如果组已经存在则成功退出，并且如果 GID 已经存在则取消-g。
– -g,--gid GID：为新组使用 GID。
– -h,--help：显示此帮助信息并退出。
– -K,--key KEY=VALUE：不使用/etc/login.defs 中的默认值。
– -o,--non-unique：允许创建有重复 GID 的组。
– -p,--password PASSWORD：为新组使用此加密过的密码。
– -r,--system：创建一个系统账户。
– -R,--root CHROOT_DIR：chroot 到的目录。

应用实例如下。

建立一个新组，并设置组 ID 加入系统。

```
# groupadd -g744 cjh
```

此时在/etc/passwd 文件中产生一个组 ID（GID）为 744 的项目。

6.7 groupdel：删除组账户

作用：用来删除组账户。

用法：groupdel 组账户名

groupdel [选项] 组

主要选项如下。

- -h,--help：显示此帮助信息并退出。
- -R,--root CHROOT_DIR：chroot 到的目录。

> **说明**
>
> 删除一个已有的用户组账户，可以使用 groupdel 命令，但是该账户不能作为所有组被其他用户引用。

应用实例如下。

用户 ab 使用组账户 bjh 作为其私有组时，不能直接删除组账户 bjh，要先删除用户 ab，再删除组账户 bjh，命令如下：

```
# userdel -r ab
# groupdel bjh
```

6.8 groupmod：修改组

作用：修改用户组的属性。

用法：groupmod [选项] 组

主要选项如下。

- -g,--gid GID：将组 ID 改为 GID。
- -h,--help：显示此帮助信息并退出。
- -n,--new-name NEW_GROUP：改名为 NEW_GROUP。
- -o,--non-unique：允许使用重复的 GID。
- -p,--password PASSWORD：将密码更改为（加密过的）PASSWORD。
- -R,--root CHROOT_DIR：chroot 到的目录。

应用实例如下。

将组 group1 的组标识号修改为 102。

```
#groupmod -g 102 group1
```

6.9 vipw：编辑/etc/passwd 文件

作用：vipw 命令可以直接编辑/etc/passwd 文件。

用法：vipw [选项]

主要选项如下。

- -g,--group：编辑 group 数据库。
- -h,--help：显示此帮助信息并退出。
- -p,--passwd：编辑 passwd 数据库。
- -q,--quiet：安静模式。
- -R,--root CHROOT_DIR：chroot 到的目录。
- -s,--shadow：编辑 shadow 或 gshadow 数据库。

说明

vipw 命令允许超级用户编辑系统密码文件/etc/passwd。默认编辑器是 vi。在对/etc/passwd 文件进行编辑时首先锁定文件，编辑完成后再解锁，这样可以保证文件的一致性。vipw 命令在功能上等同于“vi /etc/passwd”命令，但是安全性更高。

应用实例如下。

使用 vipw。

```
#vipw
```

6.10 vigr：编辑/etc/group 文件

作用：vigr 命令可以直接编辑/etc/group 文件。

用法：vigr [选项]

主要选项如下。

- -g,--group：编辑 group 数据库。
- -h,--help：显示此帮助信息并退出。
- -p,--passwd：编辑 passwd 数据库。
- -q,--quiet：安静模式。
- -R,--root CHROOT_DIR：chroot 到的目录。
- -s,--shadow：编辑 shadow 或 gshadow 数据库。

说明

vigr 通常允许超级用户编辑系统密码文件/etc/group。默认编辑器是 vi。在对/etc/group 文件进行编辑时首先锁定文件，编辑完成后再解锁，这样可以保证文件的一致性。vigr 命令在功能上等同于“vi /etc/group”命令，但是安全性更高。

应用实例如下。

使用 vigr。

```
#vigr
```

6.11 newgrp：转换组

作用：newgrp 命令用来转换用户的当前组到指定的组账户，用户必须属于该组才能执行。

用法：newgrp [-] [组]

> **说明**
>
> 如果一个用户同时属于多个用户组，那么用户可以在用户组之间切换，以便具有其他用户组的权限。用户可以在登录后，使用命令 newgrp 切换到其他用户组，这个命令的参数就是目的用户组。

应用实例如下。

```
# newgrp root
```

该命令将当前用户切换到 root 用户组，前提条件是 root 用户组确实是该用户的主组或附加组。

6.12 groups：显示组

作用：groups 命令显示指定用户所属的组；如果没有指定用户，则显示当前用户所属的组。

用法：groups [选项]... [用户名]...

显示每个输入的用户名所在的全部组，如果没有指定用户名则默认为当前进程用户（当用户组数据库发生变更时可能导致差异）。

主要选项如下。

- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

应用实例如下。

```
# groups root
root : root bin daemon sys adm disk wheel
```

该命令显示 root 用户属于 7 个组（root, bin, daemon, sys, adm, disk, wheel）。

6.13 gpasswd：添加组

作用：gpasswd 命令用来添加用户到组。

用法：gpasswd [选项] 组

主要选项如下。

- -a,--add USER：向组 GROUP 中添加用户 USER。
- -d,--delete USER：从组 GROUP 中添加或删除用户。
- -h,--help：显示此帮助信息并退出。
- -Q,--root CHROOT_DIR：要 chroot 进的目录。
- -r,--remove-password：移除组 GROUP 的密码。
- -R,--restrict：向其成员限制访问组 GROUP。
- -M,--members USER,...：设置组 GROUP 的成员列表。
- -A,--administrators ADMIN,...：设置组的管理员列表。

除非使用-A 或-M 选项，否则不能结合使用这些选项，包括-a、-d、-Q、-r、-R 几个选项。

应用实例如下。

（1）设定 user1 用户为 user 组管理员。

```
# gpasswd -A user1 user
```

（2）若要取消 user1 用户为 user 组管理员，可以设定组管理员为空。

```
# gpasswd -A user
```

6.14 whoami：显示当前用户名称

作用：显示当前用户名称。

用法：whoami [选项]...

显示与当前的有效用户 ID 相关联的用户名。与 id -un 相同。

主要选项如下。

- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

6.15 who：显示登录用户

作用：who 命令用来显示系统中有哪些用户登录系统及其修改信息。

用法：who [选项]... [文件 | 参数 1 参数 2]

显示当前已登录的用户信息。

主要选项如下。

- -a,--all：等于-b -d --login -p -r -t -T -u 选项的组合。
- -b,--boot:上次系统启动时间。
- -d,--dead：显示已死的进程。
- -H,--heading：输出头部的标题列。
- -l,--login：显示系统登录进程。
- --lookup：尝试通过 DNS 查验主机名。
- -m：只面对和标准输入有直接交互的主机和用户。
- -p,--process：显示由 init 进程衍生的活动进程。
- -q,--count：列出所有已登录用户的登录名与用户数量。
- -r,--runlevel：显示当前的运行级别。
- -s,--short：只显示名称、线路和时间（默认）。
- -T,-w,--mesg：用+、-或?标注用户消息状态。
- -u,--users：列出已登录的用户。
- --message：等于-T。
- --writable：等于-T。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

说明

who 命令能最简单、迅速地显示登录系统的用户现况。

应用实例如下。

显示登录用户名称。

```
# who
goodcjh2 tty1 2008-08-08 07:04
root pts/1 2008-08-08 09:31 (192.168.40.11)
```

6.16 id：显示用户信息

作用：id 命令用来显示用户当前的 GID、UID、用户所属的组列表。

用法：id [选项]... [用户名称]

主要选项如下。

- -a：忽略，仅为与其他版本相兼容而设计。
- -Z,--context：仅显示当前用户的安全环境。

- -g,--group：仅显示有效的用户组 ID。
- -G,--groups：显示所有组的 ID。
- -n,--name：显示组名称而非数字，可与-ugG 一起使用。
- -r,--real：显示真实 ID 而非有效 ID，与-ugG 一起使用。
- -u,--user：仅显示有效用户 ID。
- --help：显示此帮助信息并退出。
- --version：显示版本信息并退出。

应用实例如下。

显示当前登录用户的情况。

```
# id
uid=0(root)gid=0(root)groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),
10(wheel)
```

6.17 su：切换身份

作用：su 命令用来让用户暂时变更登录的身份。

用法：su[选项] [-] [USER[参数]...]

将有效用户 ID 和组 ID 更改为 USER 的 ID。单个-视为-l。如果未指定 USER，将假定为 root。

主要选项如下。

- -m,-p,--preserve-environment：不重置环境变量。
- -g,--group<组>：指定主组。
- -G,--supp-group<组>：指定一个辅助组。
- -,-l,--login：使 Shell 成为登录 Shell。
- -c,--command<命令>：使用-c 向 Shell 传递一条命令。
- --session-command<命令>：使用-c 向 Shell 传递一条命令而不创建新会话。
- -f,--fast：向 Shell 传递-f 选项（csh 或 tcsh）。
- -s,--Shell<Shell>：若/etc/Shells 允许，则运行 Shell。
- -h,--help：显示此帮助并退出。
- -V,--version：输出版本信息并退出。

应用实例如下。

将超级用户转变成一般用户。因为超级用户对系统拥有绝对的管理权限，所以使用 su 命令时，不需要输入密码。

```
#su -c /root/bin/w3bak webadmin root
```

该命令使超级用户转换成一般使用者 webadmin，并执行/root/bin/w3bak 这个 Shell 脚本，然后恢复超级用户。

6.18 pwck：检测账户

作用：pwck 命令用于检测用户账号信息的完整性。

用法：pwck [选项] [passwd [shadow]]

主要选项如下。

- -h,--help：显示此帮助信息并退出。
- -q,--quiet：只报告错误。
- -r,--read-only：显示错误和警告，但不改变文件。
- -R,--root CHROOT_DIR：chroot 到的目录。
- -s,--sort：通过 UID 排序项目。

> **说明**
>
> pwck 命令检测文件/etc/passwd 和/etc/shadow 的每一行字段的用法及数值是否正确。

应用实例如下。

验证存在于用户数据库中的所有用户和管理员，并报告（但不修正）任何错误。

```
# pwck /etc/passwd
用户 "hacluster"：目录 /home/hacluster 不存在
用户 "avahi"：目录 /var/run/avahi-daemon 不存在
用户 "avahi-autoipd"：目录 /var/lib/avahi-autoipd 不存在
用户 "memcached"：目录 /run/memcached 不存在
用户 "pulse"：目录 /var/run/pulse 不存在
用户 "gnome-initial-setup"：目录 /run/gnome-initial-setup/ 不存在
用户 "oprofile"：目录 /var/lib/oprofile 不存在
用户 "vboxadd"：目录 /var/run/vboxadd 不存在
pwck：无改变
```

6.19 grpck：检测用户组账号信息的完整性

作用：grpck 命令用于检测用户组账号信息的完整性。

使用权限：超级用户。

用法：grpck [选项] [组　[gshadow]]

主要选项如下。

- -h, --help：显示此帮助信息并退出。
- -r, --read-only：显示错误和警告，但不改变文件。

– -R, --root CHROOT_DIR：chroot 到的目录。
– -s, --sort：通过 UID 排序项目。

> **说明**
>
> grpck 命令检测文件/etc/group 和/etc/ gshadow 的每一行字段的用法及数值是否正确。

应用实例如下。

验证存在于组账号文件的所有用户组，并报告（但不修正）任何错误。

```
#grpck /etc/group
```

6.20 chsh：设置 Shell

作用：更改使用者的 Shell 设定。

使用权限：所有使用者。

用法：chsh [选项] [用户名]

主要选项如下。

– -s,--shell<shell>：指定登录 Shell。
– -l,--list-shells：打印 Shell 列表并退出。
– -u,--help：显示此帮助并退出。
– -v,--version：输出版本信息并退出。

> **说明**
>
> 每位用户在登录系统时，都会拥有预设的 Shell 环境，这个命令可更改其预设值。若不指定任何参数与用户名称，则 chsh 会以应答的方式进行设置。

应用实例如下。

Red Hat Linux 默认的 Shell 是 bash，更换 Shell 类型可以使用 chsh 命令。先输入账户密码，然后输入新 Shell 类型。如果操作正确，系统会显示“shell change”。其界面一般如下：

```
# Changing fihanging shell for cao
Password:
New shell [/bin/bash]: /bin/tcsh
```

在上述代码中，[]内是目前使用的 Shell。普通用户只能修改自己的 Shell，超级用户可以修改全体用户的 Shell。

要想查询系统提供了哪些 Shell，可以使用“chsh -l”命令。

```
# chsh -l
/bin/sh
/bin/bash
/sbin/nologin
/bin/tcsh
/bin/csh
/bin/ksh
```

6.21 chfn：修改用户信息

作用：chfn 命令用于修改用户信息。

用法：chfn [选项] [用户名]

主要选项如下。

- -f,--full-name<全名>：真实姓名。
- -o,--office<办公>：办公号码。
- -p,--office-phone<电话>：办公电话。
- -h,--home-phone<电话>：住宅电话。
- -u,--help：显示此帮助并退出。
- -v,--version：输出版本信息并退出。

说明

chfn 命令用来改变使用 finger 命令登录系统时的信息。Linux finger 命令会显示 4 段信息，包括真实姓名、工作地点、电话、家庭电话。每一段信息在命令行上指定。如果没有给定信息，chfn 将进入交谈模式，交谈模式下 chfn 命令会在每一段前提示。在提示状态下，可输入新的信息或按 Enter 键表示不修改。输入“none”则表示信息内容空白。

应用实例如下。

```
# chfn
Changing finger information for root.
Name [cjha]: cjhk
Office [2004dj]: 2004od
Office Phone [010-69245667]: 10-67543345
Home Phone [010-667853412]:
```

修改后的内容存放在/etc/passwd 文件中。

6.22 ac：显示用户在线时间的统计信息

作用：ac 命令用于显示基于当前的/var/log/wtmp 文件中的登录和退出时间输出一个关于连接时间（以小时为单位）的报告，并且输出一个总计时间。

用法：ac[选项]

主要选项如下。

- -d, --daily-totals：为每天的输出输出一个总计时间，而不是在结尾输出一个大的总计。
- -p, --individual-totals：为每个用户输出总计时间，并在最后追加一个所有用户的总计时间的累计值。
- [用户列表]：输出的是在用户列表中包括的所有用户的连接时间的总和值。用户列表由空格分隔，其中不允许有通配符。
- -f, --file filename：从指定文件而不是系统的/var/log/wtmp 文件中读取记账信息。
- -complain：当/var/log/wtmp 存在问题时（时间扭曲，丢失记录，或其他任何问题），输出一个适当的错误信息。
- --reboots：重新引导（reboot）记录不是在系统重新引导时写的，而是在系统重新启动（restart）时写的，所以不可能知道重新引导的精确的发生时间。用户在系统重新引导时可能已经在系统上登录了。
- --supplants：有时，注销记录没有写出明确的终端，因而最近的用户的自然增长时间就不能被计算。如果打算包括在一个终端上的从用户登录到下一次登录的时间（尽管可能是不正确的），就应包括此选项。
- --timewarps：一些时候，在@WTMP_FILE_LOC 文件中的记录可能突然跳回到了以前时间却没有时钟更改记录出现。在这种情况发生时，不可能知道用户登录了多长时间。如果打算依据用户的要求统计从登录到时间扭曲之间的时间，就应包括此选项。
- --compatibility：上面 3 种选项的速写，就不用敲 3 次键盘了。
- -a, --all-days：如果在输出日总计时使用了此参数，则输出每天的记录，而不是忽略没有登录活动的间隔日。没有此选项时，在这些间隔日期间自然增长的时间被列在下一天（即有登录活动的那一天）的底下。
- --tw-leniency num：设置时间扭曲的宽限为 num 秒。在/var/log/wtmp 文件中的记录可能轻微地乱了次序（最显著的是当两个登录发生在一前一后的时期，第二个可能先写了记录）。默认值被设置为 60。
- --tw-suspicious num：设置时间扭曲的不信任值为 num 秒。结果/var/log/wtmp 文件中的两个记录超出了这个秒间隔数，那么在@WTMP_FILE_LOC 文件中一定存在问题（或者用户的机器已经一年没有使用了）。
- -y, --print-year：在显示日期的时候输出年份。
- -z, --print-zeros：一个任意类别的总计（除了全部总计）是零，还是输出此总计。
- --debug：输出冗余的内部（调试）信息。

- -V, --version：在标准输出上输出版本号并退出。
- -h, --help：在标准输出上输出使用方法并退出。

应用实例如下。

（1）显示所有用户登录时间。

```
# ac -p
cjh2.97
root57.61
total 60.59
```

（2）显示 cjh 用户所有的登录时间（包括年份）。

```
# ac -d-y -p cjh
cjh2.08
Mar 25 2010 total2.08
cjh0.89
Today total0.89
```

6.23 grpconv：开启群组的投影密码

作用：grpconv 命令能开启群组的投影密码。

用法：grpconv [选项]

主要选项如下。

- -h,--help：显示此帮助信息并退出。
- -R,--root CHROOT_DIR：chroot 到的目录。

> **说明**
>
> Linux 系统里的用户和群组密码分别存放在/etc 目录下的 passwd 和 group 文件中。因系统运作所需，任何人都可以读取它们，从而造成安全上的破绽。投影密码将文件内的密码改存在/etc 目录下的 shadow 和 gshadow 文件内，只允许系统管理者读取，同时把原密码置换为“x”字符。投影密码的功能可随时开启或关闭，只需执行 grpconv 命令就能开启群组投影密码。

应用实例如下。

开启群组的投影密码。

```
# grpconv
```

6.24 grpunconv：关闭群组的投影密码

作用：执行 grpunconv 命令可关闭群组的投影密码，它会把密码从 gshadow 文件内回存到 group 文件里。

用法：grpunconv [选项]

主要选项如下。

- -h,--help：显示此帮助信息并退出。
- -R,--root CHROOT_DIR：chroot 到的目录。

应用实例如下。

关闭群组的投影密码。

```
# grpunconv
```

6.25 lastlog：显示最近登录用户的用户名、登录端口和登录时间

作用：lastlog 命令用于显示最近登录用户的用户名、登录端口和登录时间。

用法：lastlog [选项]

主要选项如下。

- -b,--before DAYS：仅打印早于 DAYS 的最近登录记录。
- -h,--help：显示此帮助信息并退出。
- -R,--root CHROOT_DIR：chroot 到的目录。
- -t,--time DAYS：仅打印晚于 DAYS 的最近登录记录。
- -u,--user LOGIN：打印 LOGIN 用户的最近登录记录。

应用实例如下。

显示最近登录用户的用户名、登录端口和登录时间。

```
]# lastlog
用户名     端口          来自                     最后登录时间
root     pts/2    192.168.118.1    五  3 月 26 11:12:16 +0800 2010
bin                                           **从未登录过**
daemon                                        **从未登录过**
adm                                           **从未登录过**
lp                                            **从未登录过**
sync                                          **从未登录过**
shutdown                                      **从未登录过**
```

```
halt                                         **从未登录过**
mail                                         **从未登录过**
rpc                                          **从未登录过**
mailnull                                     **从未登录过**
smmsp                                        **从未登录过**
…
```

6.26 logname：显示当前用户的名称

作用：logname 命令用于显示当前用户的名称。

用法：logname [选项]

主要选项如下。

- --help：显示此帮助信息并退出。
- --version：输出版本信息并退出。

应用实例如下。

显示当前用户的名称。

```
# logname
root
```

6.27 users：显示当前登录到系统的用户

作用：users 命令用于显示当前登录到系统的用户。

用法：users [选项]... [文件]

主要选项如下。

- --help：显示此帮助信息并退出。
- --version：输出版本信息并退出。

应用实例如下。

显示当前登录到系统的用户。

```
# users
root root root
```

6.28 lastb：显示登录系统失败用户的相关信息

作用：列出登录系统失败用户的相关信息。记录的登录失败的用户名单全部显示出来。

用法：lastb [-adRx] [-f <记录文件>] [-n <显示列数>] [账号名称...] [终端机编号...]

主要选项如下。

- -a：把从何处登录系统的主机名称或 IP 地址显示在最后一行。
- -f<记录文件>：指定记录文件。
- -n<显示列数>或-<显示列数>：设置列出名单的显示列数。
- -x：显示系统关机、重新开机，以及执行等级的改变等信息。
- -n：显示最近登录信息的行数。
- -t：根据登录的时间显示指定用户的登录状态。
- -R：简略显示主机名区域。
- -d：对于非本地登录的用户，将显示的 IP 地址转换为主机名。
- -i：类似于“-d”，将显示的主机名称转换为 IP 地址。
- -o：读取旧的类型的 wtmp 文件。
- -x：显示系统登录等级。
- --help：显示帮助信息。
- --version：显示版本信息。

> **说明**
>
> 单独执行 lastb 指令，它会读取位于/var/log 目录下名称为 btmp 的文件，并把该文件内容列出。

应用实例如下。

（1）显示最近 5 个登录失败记录。

```
# lastb -n 5
        pts/2   192.168.118.132  Fri Mar 26 10:09 - 10:09  (00:00)
        pts/2   192.168.118.132  Fri Mar 26 10:09 - 10:09  (00:00)
        pts/2   192.168.118.132  Fri Mar 26 10:09 - 10:09  (00:00)
cjh     pts/2   192.168.118.132  Fri Mar 26 10:09 - 10:09  (00:00)
w       pts/2   192.168.118.132  Fri Mar 26 09:34 - 09:34  (00:00)
btmp begins Thu Mar 25 20:33:50 2010
```

（2）显示 2010-04-01 前的最近 3 个登录失败记录。

```
# lastb -t 20100401000000 -n 3
        pts/2   192.168.118.132  Fri Mar 26 10:09 - 10:09  (00:00)
```

```
         pts/2   192.168.118.132  Fri Mar 26 10:09 - 10:09  (00:00)
         pts/2   192.168.118.132  Fri Mar 26 10:09 - 10:09  (00:00)

btmp begins Thu Mar 25 20:33:50 2010
root  ssh:notty 192.168.118.1    Thu Mar 25 23:43 - 23:43  (00:00)
root  ssh:notty 192.168.118.1    Thu Mar 25 20:33 - 20:33  (00:00)

btmp begins Thu Mar 25 20:33:50 2014
```

07 Linux 的备份和压缩命令

7.1 tar：备份文件

作用：tar 是用来建立、还原备份文件的工具程序，它可以加入、解开备份文件内的文件。

用法：tar [选项] [文件]

1．主操作模式

- -A,--catenate,--concatenate：追加 tar 文件至归档。
- -c,--create：创建一个新归档。
- -d,--diff,--compare：找出归档和文件系统的差异。
- --delete：从归档（非磁带）中删除。
- -r,--append：追加文件至归档结尾。
- -t,--list：列出归档内容。
- --test-label：测试归档卷标并退出。
- -u,--update：仅追加比归档中副本更新的文件。
- -x,--extract,--get：从归档中解出文件。

2．操作修饰符

- --check-device：当创建增量归档时检查设备号（默认）。
- -g,--listed-incremental=FILE：处理新式的 GNU 格式的增量备份。
- -G,--incremental：处理老式的 GNU 格式的增量备份。
- --ignore-failed-read：当遇上不可读文件时不要以非零值退出。
- --level=NUMBER：所创建的增量列表归档的输出级别。
- -n,--seek：归档可检索。
- --no-check-device：当创建增量归档时不要检查设备号。
- --no-seek：归档不可检索。
- --occurrence[=NUMBER]：仅处理归档中每个文件的第 NUMBER 个事件；仅当与子命令--delete、--diff、--extract 或--list 中的一个联合使用时，此选项才有效，而且不管文件列表是以命令行形式给出还是通过-T 选项指定的；NUMBER 值默认为 1。
- --sparse-version=MAJOR[.MINOR]：设置所用的离散格式版本（隐含--sparse）。

- -S,--sparse：高效处理离散文件。

3．重写控制

- -k,--keep-old-files：解压时不要替换存在的文件。
- --keep-newer-files：不要替换比归档中副本更新的已存在的文件。
- --no-overwrite-dir：保留已存在目录的元数据。
- --overwrite：解压时重写存在的文件。
- --overwrite-dir：解压时重写已存在目录的元数据（默认）。
- --recursive-unlink：解压目录之前先清除目录层次。
- --remove-files：在添加文件至归档后删除它们。
- --skip-old-files：解压时不要替换存在的文件。
- -U,--unlink-first：在解压要重写的文件之前先删除它们。
- -W,--verify：在写入以后尝试校验归档。

4．选择输出流

- --ignore-command-error：忽略子进程的退出代码。
- --no-ignore-command-error：将子进程的非零退出代码认为发生错误。
- -O,--to-stdout：解压文件至标准输出。
- --to-command=COMMAND：将解压的文件通过管道传送至另一个程序。

5．操作文件属性

- --atime-preserve[=METHOD]：在输出的文件上保留访问时间，要么通过在读取(默认 METHOD='replace')后还原时间，要么在第一次(METHOD='system')设置时间。
- --delay-directory-restore：直到解压结束才设置修改时间和所解目录的权限。
- --group=名称：强制将 NAME 作为所添加的文件的组所有者。
- --mode=CHANGES：强制将所添加的文件（符号）更改为权限 CHANGES。
- --mtime=DATE-OR-FILE：从 DATE-OR-FILE 中为添加的文件设置 mtime。
- -m,--touch：不要解压文件的修改时间。
- --no-delay-directory-restore：取消--delay-directory-restore 选项的效果。
- --no-same-owner：将文件解压为用户自己所有（普通用户默认此项）。
- --no-same-permissions：从归档中解压权限时使用用户的掩码位（默认为普通用户服务）。
- --numeric-owner：总是以数字代表用户/组的名称。
- --owner=名称：强制将 NAME 作为所添加的文件的所有者。
- -p,--preserve-permissions,--same-permissions：解压文件权限信息（默认只为超级用户服务）。
- --preserve：与-p 和-s 一样。
- --same-owner：尝试解压时保持所有者关系一致（超级用户默认此项）。
- -s,--preserve-order,--same-order：为解压至匹配归档排序名称。

6．设备选择和切换

- -f,--file=ARCHIVE：使用归档文件或 ARCHIVE 设备。

- --force-local：即使归档文件存在副本，还是把它认为是本地归档。
- -F,--info-script=名称,--new-volume-script=名称：在每卷磁带最后运行脚本(隐含-M)。
- -L,--tape-length=NUMBER：写入 NUMBER×1024 字节后更换磁带。
- -M,--multi-volume：创建/列出/解压多卷归档文件。
- --rmt-command=COMMAND：使用指定的 rmtCOMMAND 代替 rmt。
- --rsh-command=COMMAND：使用远程 COMMAND 代替 rsh。
- --volno-file=FILE：使用/更新 FILE 中的卷数。

7．设备分块

- -b,--blocking-factor=BLOCKS：每个记录 BLOCKS×512 字节。
- -B,--read-full-records：读取时重新分块（只对 4.2BSD 管道有效）。
- -i,--ignore-zeros：忽略归档中的零字节块（即文件结尾）。
- --record-size=NUMBER：每个记录的字节数 NUMBER×512。

8．选择归档格式

- -H,--format=FORMAT：创建指定格式的归档。FORMAT 是以下格式中的一种。
  - Ø gnu：GNU tar 1.13.x 格式。
  - Ø oldgnu：GNU 格式 as per tar<=1.12。
  - Ø pax：POSIX 1003.1-2001(pax)格式。
  - Ø posix：等同于 pax。
  - Ø ustar：POSIX 1003.1-1988(ustar)格式。
  - Ø v7：old V7 tar 格式。
- --old-archive,--portability：等同于--format=v7。
- --pax-option=关键字[[:]=值] [,关键字[[:]=值]]..：控制 pax 关键字。
- --posix：等同于--format=posix。
- -V,--label=TEXT：创建带有卷名 TEXT 的归档；在列出/解压时，使用 TEXT 作为卷名的模式串。

9．压缩选项

- -a,--auto-compress：使用归档扩展名来决定压缩程序。
- -I,--use-compress-program=PROG：通过 PROG 过滤（必须是能接受-d 选项的程序）。
- -j,--bzip2：通过 bzip2 过滤归档。
- -J,--xz：通过 xz 过滤归档。
- --lzip：通过 lzip 过滤归档。
- --lzma：通过 lzma 过滤归档。
- --lzop：通过 lzop 过滤文件。
- --no-auto-compress：不使用归档扩展名来决定压缩程序。
- -z,--gzip,--gunzip,--ungzip：通过 gzip 过滤归档。
- -Z,--compress,--uncompress：通过 compress 过滤归档。

10．本地文件选择

- --add-file=FILE：添加指定的 FILE 至归档（如果名字以-开始会很有用的）。
- --backup[=CONTROL]：在删除前备份，选择 CONTROL 版本。
- -C,--directory=DIR：改变至目录 DIR。
- --exclude=PATTERN：排除以 PATTERN 指定的文件。
- --exclude-backups：排除备份和锁文件。
- --exclude-caches：除标识文件本身外，排除包含 CACHEDIR.TAG 的目录中的内容。
- --exclude-caches-all：排除包含 CACHEDIR.TAG 的目录。
- --exclude-caches-under：排除包含 CACHEDIR.TAG 的目录中的所有内容。
- --exclude-tag=FILE：除 FILE 自身外，排除包含 FILE 的目录中的内容。
- --exclude-tag-all=FILE：排除包含 FILE 的目录。
- --exclude-tag-under=FILE：排除包含 FILE 的目录中的所有内容。
- --exclude-vcs：排除版本控制系统目录。
- -h,--dereference：跟踪符号链接；将它们所指向的文件归档并输出。
- --hard-dereference：跟踪硬链接；将它们所指向的文件归档并输出。
- -K,--starting-file=MEMBER-NAME：从归档中的 MEMBER-NAME 成员处开始。
- --newer-mtime=DATE：当只有数据改变时比较数据和时间。
- --no-null：禁用上一次的效果。
- --no-recursion：避免目录中的自动降级。
- --no-unquote：不以-T 读取的文件名作为引用结束。
- --null：-T 读取以空终止的名字，-C 禁用。
- -N,--newer=DATE-OR-FILE,--after-date=DATE-OR-FILE：只保存比 DATE-OR-FILE 更新的文件。
- --one-file-system：创建归档时保存在本地文件系统中。
- -P,--absolute-names：不要从文件名中清除引导符“/”。
- --recursion：目录递归（默认）。
- --suffix=STRING：在删除前备份，除非被环境变量 SIMPLE_BACKUP_SUFFIX 覆盖，否则覆盖常用后缀（‘’）。
- -T,--files-from=FILE 从 FILE：中获取文件名来解压或创建文件。
- --unquote：以-T 读取的文件名作为引用结束（默认）。
- -X,--exclude-from=FILE：排除 FILE 中列出的模式串。

11．文件名变换

- --strip-components=NUMBER：解压时从文件名中清除 NUMBER 个引导部分。
- --transform=EXPRESSION,--xform=EXPRESSION：使用 sed 代替 EXPRESSION 来进行文件名变换。

12．文件名匹配选项（同时影响排除和包括模式串）

- --anchored：模式串匹配文件名头部。

- --ignore-case：忽略大小写。
- --no-anchored：模式串匹配任意“/”后的字符（默认对 exclusion 有效）。
- --no-ignore-case：匹配大小写（默认）。
- --no-wildcards：逐字匹配字符串。
- --no-wildcards-match-slash：通配符不匹配“/”。
- --wildcards：使用通配符（默认对 exclusion 有效）。
- --wildcards-match-slash：通配符匹配“/”（默认对排除操作有效）。

13．提示性输出

- --checkpoint[=NUMBER]：每隔 NUMBER 个记录显示进度信息（默认为 10 个）。
- --checkpoint-action=ACTION：在每个检查点上执行 ACTION。
- --full-time：打印输出文件的时候使用全称。
- --index-file=FILE：将详细输出发送至 FILE。
- -l,--check-links：只要不是所有链接都被输出就打印信息。
- --no-quote-chars=STRING：禁用来自 STRING 的字符引用。
- --quote-chars=STRING：来自 STRING 的额外的引用字符。
- --quoting-style=STYLE：设置名称引用风格；有效的 STYLE。
- -R,--block-number：每个信息都显示归档内的块数。
- --show-defaults：显示 tar 默认选项。
- --show-omitted-dirs：列表或解压时，列出每个不匹配查找标准的目录。
- --show-transformed-names,--show-stored-names：显示变换后的文件名或归档名。
- --totals[=SIGNAL]：处理归档后打印出总字节数；当此 SIGNAL 被触发时带参数“-”打印总字节数；允许的信号为 SIGHUP、SIGQUIT、SIGINT、SIGUSR1 和 SIGUSR2；同时也接受不带 SIG 前缀的信号名称。
- --utc：以 UTC 格式打印文件修改时间。
- -v,--verbose：详细地列出处理的文件。
- --warning=KEYWORD：警告控制。
- -w,--interactive,--confirmation：每次操作都要求确认。

14．兼容性选项

-o：创建归档时，相当于--old-archive；展开归档时，相当于--no-same-owner。

15．其他选项

- -?,--help：显示此帮助列表。
- --restrict：禁用某些潜在的有危险的选项。
- --usage：显示简短的用法说明。
- --version：打印程序版本。

长选项和相应短选项具有相同的强制参数或可选参数。

应用实例如下。

（1）跨越多张软盘备份系统/usr/local 下的所有内容，同时进行写入校验。

```
# tar -cWMf /dev/fd0 /usr/local
```

（2）备份/目录下自 04/11/2006 改过的文件到磁带设备/dev/st0 中。

```
# tar -cvf /dev/st0 -N 04/11/2006
```

（3）保持文件权限，备份目录（不包括/proc）到磁带设备/dev/st0 中。

```
# tar -cpf /dev/st0 / --exclude=/proc
```

（4）恢复备份到目录。

```
# tar -xpf /dev/st0 -C /
```

（5）从备份文件恢复指定的文件 etc/passwd。

```
# tar -xpf /dev/st0 etc/passwd
```

（6）做一个完全备份。

```
# tar -zcvf /backup/bp_full.tar.gz /
```

（7）每隔 7 天执行一次增量备份。

```
# tar -c -T /tmp/filelist -f /backup/bp_add.tar.gz
```

7.2 dump：备份文件系统

作用：dump 为备份工具程序，可将目录或整个文件系统备份至指定的设备，或备份成一个大文件。

用法：dump [选项] [-0123456789] [-b <区块大小>] [-B <区块数目>] [-d <密度>] [-f <设备名称>] [-h <层级>] [-s <磁带长度>] [-T <日期>] [目录或文件系统]

dump [-wW]

主要选项如下。

- -0123456789：备份的层级。
- -b<区块大小>：指定区块的大小，单位为 KB。
- -B<区块数目>：指定备份卷册的区块数目。
- -c：修改备份磁带预设的密度与容量。
- -d<密度>：设置磁带的密度，单位为 BPI。
- -f<设备名称>：指定备份设备。
- -h<层级>：当备份层级等于或大于指定的层级时，将不备份用户标识为“nodump”的文件。
- -n：当备份工作需要管理员介入时，向所有“operator”群组中的使用者发出通知。
- -s<磁带长度>：备份磁带的长度，单位为英尺。
- -T<日期>：指定开始备份的时间与日期。

- -u：备份完毕后，在/etc/dumpdates 中记录备份的文件系统、层级、日期与时间等。
- -w：与-W 类似，但仅显示需要备份的文件。
- -W：显示需要备份的文件及其最后一次备份的层级、时间与日期。

应用实例如下。

执行一个完全 Linux 备份，备份所有 Ext2 和 Ext3 文件系统到一个 SCSI 磁带设备中。

```
#dump 0f /dev/nst0 /boot
#dump 0f /dev/nst0 /
```

在这个例子中，系统中有两个文件系统，分别用于/boot 和/。这是常见的配置，它们必须在执行备份时单独地引用。/dev/nst0 引用第一台 SCSI 磁带驱动器，不过以非重绕模式引用，这样可以确保各个卷在磁带上逐个排列。dump 的一个有趣特性是其内置的增量备份功能，在上面的例子中，0 表示 0 级或基本级备份。这是完全系统备份，应定期执行以保存整个系统。对于后续的备份，可以使用其他数字（1～9）来代替 0，以改变备份级别。1 级备份会保存自从执行 0 级备份以来更改过的所有文件，2 级备份会保存自从执行 1 级备份以来更改过的所有文件，依此类推。使用 tar 和脚本可以执行相同的功能，但要求脚本创建人员提供一种机制来确定上次备份是何时执行的。dump 具有自己的机制，即在执行备份时会输出一个更新文件（/etc/dumpupdates）。这个更新文件将在每次执行 0 级备份时被重设，后续级别的备份会保留其标记，直至执行另一次 0 级备份。如果正在执行基于磁带的备份，dump 会自动跟踪多个卷。

7.3 cpio（copy in/out）：建立、还原备份文件

作用：cpio 是用来建立、还原备份档的工具程序，它可以加入、解开 cpio 或 tra 备份档内的文件。

用法：cpio [选项] [目的目录]

主要选项如下。

- -0 或--null：接受新增列控制字符，通常配合 find 指令的“-print0”参数使用。
- -a 或--reset-access-time：重新设置文件的存取时间。
- -A 或--append：附加到已存在的备份档中，且这个备份档必须存放在磁盘上，而不能放置于磁带机里。
- -b 或--swap：此参数的效果和同时指定“-sS”参数相同。
- -B：将输入/输出的区块大小改成 5210B。
- -c：使用旧 ASCII 备份用法。
- -C<区块大小>或--io-size=<区块大小>：设置输入/输出的区块大小，单位是 Byte。
- -d 或--make-directories：如有需要 cpio 会自行建立目录。
- -E<范本文件>或--pattern-file=<范本文件>：指定范本文件，其内含一个或多个范本样式，让 cpio 解开符合范本条件的文件，用法为每列一个范本样式。
- -f 或--nonmatching：让 cpio 解开所有不符合范本条件的文件。

- -F<备份档>或--file=<备份档>：指定备份档的名称，用来取代标准输入或输出，也能借此通过网络使用另一台主机的保存设备存取备份档。
- -H<备份用法>：指定备份时欲使用的文件用法。
- -i 或--extract：执行 copy-in 模式，还原备份档。
- -I<备份档>：指定备份档的名称，用来取代标准输入，也能借此通过网络使用另一台主机的保存设备读取备份档。
- -k：此参数将忽略不予处理，仅负责解决 cpio 不同版本间的兼容性问题。
- -l 或--link：以硬链接的方式取代复制文件，可在 copy-pass 模式下运用。
- -L 或--dereference：不建立符号链接，直接复制该链接所指向的原始文件。
- -m 或 preserve-modification-time：不去更换文件的更改时间。
- -M<回传信息>或--message=<回传信息>：设置更换保存媒体的信息。
- -n 或--numeric-uid-gid：使用"-tv"参数列出备份档的内容时，若再加上参数"-n"，则会以用户识别码和群组识别码替代拥有者及群组名称列出文件清单。
- -o 或--create：执行 copy-out 模式，建立备份档。
- -O<备份档>：指定备份档的名称，用来取代标准输出，也能借此通过网络使用另一台主机的保存设备存放备份档。
- -p 或--pass-through：执行 copy-pass 模式，略过备份步骤，直接将文件复制到目的目录。
- -r 或--rename：当有文件名称需要改动时，采用互动模式。
- -R<拥有者><:/.><所属群组>或----owner<拥有者><:/.><所属群组>：在 copy-in 模式下还原备份档；或在 copy-pass 模式下复制文件时，可指定这些备份复制的文件的拥有者与所属群组。
- -s 或--swap-bytes：交换每对字节的内容。
- -S 或--swap-halfwords：交换每半个字节的内容。
- -t 或--list：将输入的内容呈现出来。
- -u 或--unconditional：置换所有文件，不论日期时间的新旧与否，皆不予询问而直接覆盖。
- -v 或--verbose：详细显示指令的执行过程。
- -V 或--dot：执行指令时，在每个文件的执行程序前面加上"."。
- --block-size=<区块大小>：设置输入/输出的区块大小。假如设置数值为 5，则区块大小为 2500；若设置为 10，则区块大小为 5120，依此类推。
- --force-local：强制将备份档存放在本地主机上。
- --help：在线帮助。
- --no-absolute-filenames：使用相对路径建立文件名称。
- --no-preserve-owner：不保留文件的拥有者，谁解开了备份档，那些文件就归谁所有。
- -only-verify-crc：当备份档采用 CRC 备份用法时，可使用这项参数检查备份档内的每个文件是否正确无误。
- --quiet：不显示复制了多少区块。
- --sparse：倘若一个文件内含有大量的连续 0 字节，则将此文件存为稀疏文件。

– --version：显示版本信息。

应用实例如下。

（1）复制当前目录中的所有文件到磁带设备/dev/rmt0 中。

```
#find . | /usr/sysv/bin/cpio -oc >/dev/rmt0
```

（2）从 cpio 归档中只换取一个常规文件中的文件列表。

```
#cat ar | /usr/sysv/bin/cpio -i -E Efile
```

（3）把/home 目录备份到 SCSI 磁带设备中。

```
#$ls /home | cpio -o > /dev/st0
```

（4）定时备份。

也可以计划 cpio 的备份，用 crontab 工具设置为基于任何时间的有规律的运行。例如，要在每晚 2∶00 把/home 目录备份到 SCSI 磁带驱动器中，则将下面的语句添加到 root 的 crontab 文件中即可。

```
00 02 * * * ls /home | cpio -o > /dev/st0
```

7.4 restore：还原备份下来的文件或整个文件系统（一个分区）

作用：restore 指令所进行的操作和 dump 指令相反，倾倒操作可用来备份文件，而还原操作则是写回这些已备份的文件。

用法：restore [选项] [-b <区块大小>] [-D <文件系统>] [-f <备份文件>] [-s <文件编号>]

restore [选项] [-b <区块大小>] [-f <备份文件>] [-s <文件编号>]

restore [选项] [-b <区块大小>] [-f <备份文件>] [-s <文件编号>]

restore [选项] [-b <区块大小>] [-D <文件系统>] [-f <备份文件>] [-s <文件编号>]

restore [选项] [-b <区块大小>] [-D <文件系统>] [-f <备份文件>] [-s <文件编号>] [文件...]

restore [选项] [-b <区块大小>] [-D <文件系统>] [-f <备份文件>] [-s <文件编号>] [文件...]

主要选项如下。

– -b<区块大小>：设置区块大小，单位是 Byte。
– -c：不检查倾倒操作的备份用法，仅允许读取使用旧用法的备份文件。
– -C：使用对比模式，将备份的文件与现行的文件相互对比。
– -D<文件系统>：允许用户指定文件系统的名称。
– -f<备份文件>：从指定的文件中读取备份数据，进行还原操作。
– -h：仅解出目录而不包括与该目录相关的所有文件。
– -i：使用互动模式，在进行还原操作时，restore 指令将依序询问用户。
– -m：解开符合指定的 inode 编号的文件或目录而非采用文件名称指定。
– -r：进行还原操作。

- -R：全面还原文件系统时，检查应从何处开始进行。
- -s<文件编号>：当备份数据超过一卷磁带时，可以指定备份文件的编号。
- -t：指定文件名称；若该文件已存在于备份文件中，则列出它们的名称。
- -v：显示指令执行过程。
- -x：设置文件名称，且从指定的存储媒体里读入它们；若该文件已存在于备份文件中，则将其还原到文件系统内。
- -y：不询问任何问题，一律以同意回答并继续执行指令。

应用实例如下。

从磁带备份中提取/etc 目录，可使用以下命令：

```
#restore -xf /dev/nst0 /etc
```

7.5 bunzip2：解压缩.bz2 文件

作用：bunzip2 命令是用于.bz2 文件的解压缩程序。

用法：bunzip2[选项] [.bz2 压缩文件]

主要选项如下。

- -f 或--force：解压缩时，若输出的文件与现有文件同名，预设不会覆盖现有的文件。若要覆盖，请使用此参数。
- -k 或--keep：在解压缩后，预设会删除原来的压缩文件。若要保留压缩文件，请使用此参数。
- -s 或--small：降低程序执行时内存的使用量。
- -v 或--verbose：解压缩文件时，显示详细的信息。
- -l,--license：显示软件版本、许可证条款及条件。
- -V 或--version：显示版本信息。

说明

bunzip2 可解压缩.bz2 用法的压缩文件。bunzip2 实际上是 bzip2 的符号链接，执行 bunzip2 与 bzip2 -d 的效果相同。

应用实例如下。

解压缩一个.bz2 文件。

```
#bunzip2 goodcjh.bz2
```

7.6 bzip2：解压缩.bz2 文件

作用：bzip2 命令是用于.bz2 文件的解压缩程序。

用法：bzip2 [-cdfhkLstvVz] [--repetitive-best] [--repetitive-fast] [-压缩等级] [要压缩的文件]

主要选项如下。

- -c 或--stdout：将压缩与解压缩的结果送到标准输出。
- -d 或--decompress：执行解压缩。
- -f 或--force：在压缩或解压缩时，若输出文件与现有文件同名，默认不会覆盖现有文件。若要覆盖，请使用此参数。
- -h 或--help：显示帮助。
- -k 或--keep：bzip2 在压缩或解压缩后会删除原始的文件。若要保留原始文件，请使用此参数。
- -s 或--small：降低程序执行时内存的使用量。
- -t 或--test：测试.bz2 压缩文件的完整性。
- -v 或--verbose：压缩或解压缩文件时，显示详细的信息。
- -z 或--compress：强制执行压缩。
- -L,--license：显示软件版本、许可证条款及条件。
- -V 或--version：显示版本信息。
- --repetitive-best：若文件中有重复出现的资料时，可利用此参数提高压缩效果。
- --repetitive-fast：若文件中有重复出现的资料时，可利用此参数加快执行速度。
- -压缩等级：压缩时的区块大小。

应用实例如下。

将/tmp/man.config 文件以 bzip2 用法压缩。

```
# bzip2 -z man.config
```

此时 man.config 会变成 man.config.bz2。

7.7 bzgrep：使用正则表达式搜索.bz2 压缩包中的文件

作用：bzgrep 命令使用正则表达式搜索.bz2 压缩包中的文件，将匹配的行显示到标准输出。

用法：bzgrep [grep_options] [files]

主要选项如下。

- grep_options：指定要搜索的模式。
- fiels：指定要搜索的.bz2 压缩包。

应用实例如下。

```
$ bzgrep -i "CONSOLE=.*"trace.bz2
```

```
2010-10-11T08:40:28.100 gs(16985): CONSOLE=/dev/pts/0
2010-10-11T08:40:29.772 gs(17031): CONSOLE=/dev/pts/0
2010-10-11T08:40:58.140 gs(17099): CONSOLE=/dev/pts/0
2010-10-11T08:41:27.547 gs(17164): CONSOLE=/dev/pts/0
2010-10-11T08:41:57.962 gs(17233): CONSOLE=/dev/pts/0
2010-10-11T08:42:28.392 gs(17294): CONSOLE=/dev/pts/0
2010-10-11T08:42:57.721 gs(17439): CONSOLE=/dev/pts/0
```

7.8 unzip：解压缩.zip 文件

作用：unzip 命令用来解压缩.zip 文件类型的文件。

用法：unzip [选项] [-P <密码>] [.zip 文件] [文件] [-d <目录>] [-x <文件>]

　　　unzip [-Z]

主要选项如下。

- -c：将解压缩的结果显示到屏幕上，并对字符做适当的转换。
- -f：更新现有的文件。
- -l：显示压缩文件内所包含的文件。
- -p：与-c 参数类似，会将解压缩的结果显示到屏幕上，但不会执行任何的转换。
- -t：检查压缩文件是否正确。
- -u：与-f 参数类似，但除了更新现有的文件外，也将压缩文件中的其他文件解压缩到目录中。
- -v：执行时显示详细的信息。
- -z：仅显示压缩文件的备注文字。
- -a：对文本文件进行必要的字符转换。
- -b：不要对文本文件进行字符转换。
- -C：压缩文件中的文件名称区分大小写。
- -j：不处理压缩文件中原有的目录路径。
- -L：将压缩文件中的全部文件名改为小写。
- -M：将输出结果送到 more 程序处理。
- -n：解压缩时不要覆盖原有的文件。
- -o：不必先询问用户，unzip 执行后覆盖原有文件。
- -P<密码>：使用 zip 的密码选项。
- -q：执行时不显示任何信息。
- -s：将文件名中的空白字符转换为底线字符。
- -V：保留 VMS 的文件版本信息。
- -X：解压缩的同时回存文件原来的 UID/GID。
- [.zip 文件]：指定.zip 压缩文件。

- [文件]：指定要处理.zip 压缩文件中的哪些文件。
- -d<目录>：指定文件解压缩后所要存储的目录。
- -x<文件>：指定不要处理.zip 压缩文件中的哪些文件。
- -Z：unzip -Z 等于执行 zipinfo 指令。

应用实例如下。

解压缩一个.zip 文件类型的文件。

```
#unzip goodcjh.zip
```

7.9 bzip2recover：修复损坏的.bz2 文件

作用：bzip2recover 命令用来修复损坏的.bz2 文件。

用法：bzip2recover[.bz2 压缩文件]

说明

bzip2 以区块的方式来压缩文件，每个区块视为独立的单位。因此，当某一区块损坏时，便可利用 bzip2recover 试着将文件中的区块隔开，以便解压缩正常的区块。通常只适用于压缩文件很大的情况。

应用实例如下。

修复损坏的.bz2 文件。

```
# bzip2recover godocjh.bz2
```

7.10 gzip：压缩文件

作用：gzip 命令用来压缩文件。

用法：gzip [选项] [-S <压缩字尾字符串>] [-<压缩效率>] [--best/fast] [文件...]

gzip [选项] [-S <压缩字尾字符串>] [-<压缩效率>] [--best/fast] [目录]

主要选项如下。

- -a 或--ascii：使用 ASCII 文字模式。
- -c 或--stdout 或--to-stdout：把压缩后的文件输出到标准输出设备，不去改动原始文件。
- -d 或--decompress 或----uncompress：解开压缩文件。
- -f 或--force：强行压缩文件，不理会文件名称或硬链接是否存在，以及该文件是否为符号链接。

– -h 或--help：在线帮助。
– -l 或--list：列出压缩文件的相关信息。
– -L 或--license：显示版本与版权信息。
– -n 或--no-name：压缩文件时，不保存原来的文件名称及时间戳。
– -N 或--name：压缩文件时，保存原来的文件名称及时间戳。
– -q 或--quiet：不显示警告信息。
– -r 或--recursive：递归处理，将指定目录下的所有文件及子目录一并处理。
– -S<压缩字尾字符串>或----suffix<压缩字尾字符串>：更改压缩字尾字符串。
– -t 或--test：测试压缩文件是否正确无误。
– -v 或--verbose：显示指令执行过程。
– -V 或--version：显示版本信息。
– -<压缩效率>：压缩效率是一个介于 1～9 之间的数值，预设值为“6”。指定越大的数值，压缩效率就会越高。
– --best：此参数的效果和指定“-9”参数相同。
– --fast：此参数的效果和指定“-1”参数相同。

说明

gzip 是一个广泛使用的压缩程序，文件经它压缩后，其名称后面会多出“.gz”扩展名。

应用实例如下。

（1）将/etc/man.config 文件以 gzip 压缩。

```
# gzip man.config
```

此时 man.config 会变成 man.config.gz。

（2）将 man.config.gz 文件解压缩。

```
# gzip -d man.config.gz
```

（3）将 man.config 文件用最佳的压缩比压缩，并保留原本的文件。

```
# gzip -9 -c man.config > man.config.gz
```

7.11 compress：压缩、解压文件

作用：compress 命令用来压缩或解压文件。

用法：compress [选项] [-b <压缩效率>] [文件或目录...]

主要选项如下。

- -c：把压缩后的文件输出到标准输出设备，不去改动原始文件。
- -d：对文件进行解压缩而非压缩。
- -f：强制保存压缩文件，不理会文件名称或硬链接是否存在、该文件是否为符号链接及压缩效率高低的问题。
- -r：递归处理，将指定目录下的所有文件及子目录一并处理。
- -v：显示指令执行过程。
- -V：显示指令版本及程序预设值。
- -b<压缩效率>：压缩效率是一个介于 9～16 之间的数值，预设值为“16”，指定越大的数值，压缩效率就越高。

说明

compress 是一个历史悠久的压缩程序，文件经它压缩后，其名称后面会多出“.Z”扩展名。当要解压缩时，可执行 uncompress 指令。事实上，uncompress 是指向 compress 的符号链接，因此不论是压缩还是解压缩，都可通过 compress 指令单独完成。compress 命令压缩数据时，利用自适应的 Lempel-Zev 编码来减小文件。每一个被参数 File 指定的原文件将被可能的压缩文件所代替，压缩文件名加扩展名.Z。压缩文件保留与原文件相同的所有权、方式和修改时间。如果指定文件的路径大于 1023 字节，则该命令不工作。如果没有指定文件，标准输入将被压缩到标准输出。如果压缩没有减小文件大小，将对标准错误写一个消息而不替换原文件。

7.12 gzexe：压缩执行文件

作用：压缩执行文件。

用法：gzexe [-d] [执行文件...]

主要选项如下。

-d：解开压缩文件。

说明

gzexe 是用来压缩执行文件的程序。当执行被压缩过的执行文件时，该文件会自动解压然后继续执行，和使用一般的执行文件相同。Linux 下有一个类似 DOS 里的 pklite 和 lzexe 的命令--gzexe。压缩率一般都可以超过 50%，在空间紧张时很有用。

应用实例如下。

（1）压缩一个执行文件。

```
#gzexe gawkgawk: 54.5%
```

（2）用 gzexe 加密 Shell。这种加密方式不是非常保险的方法，但是能够满足一般的加密用途，可以隐蔽脚本中的密码等信息。它使用系统自带的 gzexe 程序，它不但加密，同时压缩文件。

```
#gzexe file.sh
```

它会把原来没有加密的文件备份为 file.sh～，同时 file.sh 即被变成加密文件。

7.13 lha：压缩或解压缩文件

作用：压缩或解压缩文件。

用法：lha [选项] [-<e/x>w=<目的目录>] [-<a/u>z] [压缩文件] [文件...]

　　　lha [选项] [-<a/u>o] [-<e/x>w=<目的目录>] [-<a/u>z] [压缩文件] [目录...]

主要选项如下。

- -a 或 a：压缩文件，并加入到压缩文件内。
- -a<0/1/2>/u</0/1/2>：压缩文件时，采用不同的文件头。
- -c 或 c：压缩文件，重新建构新的压缩文件后，再将其加入。
- -d 或 d：从压缩文件内删除指定的文件。
- -<a/c/u>d 或<a/c/u>d：压缩文件，然后将其加入，重新建构，更新压缩文件或删除原始文件，也就是把文件移到压缩文件中。
- -e 或 e：解开压缩文件。
- -f 或 f：强制执行 lha 命令，在解压时会直接覆盖已有的文件而不加以询问。
- -g 或 g：使用通用的压缩用法，便于解决兼容性的问题。
- -<e/x>i 或<e/x>I：解开压缩文件时，忽略保存在压缩文件内的文件路径，直接将其解压后存放在现行目录下或指定的目录中。
- -l 或 l：列出压缩文件的相关信息。
- -m 或 m：此参数的效果和同时指定“-ad”参数相同。
- -n 或 n：不执行指令，仅列出实际执行时会进行的动作。
- -<a/u>o 或<a/u>o：采用 lharc 兼容用法，将压缩后的文件加入，更新压缩文件。
- -p 或 p：从压缩文件内输出到标准输出设备。
- -q 或 q：不显示指令执行过程。
- -t 或 t：检查备份文件内的每个文件是否正确无误。
- -u 或 u：更换较新的文件到压缩文件内。
- -u</0/1/2>或 u</0/1/2>：在文件压缩时采用不同的文件头，然后更新到压缩文件内。

- -v 或 v：详细列出压缩文件的相关信息。
- -<e/x>w=<目的目录>或<e/x>w=<目的目录>：指定解压缩的目录。
- -x 或 x：解开压缩文件。
- -<a/u>z 或<a/u>z：不压缩文件，直接把它加入，更新压缩文件。

说明

lha 是从 lharc 演变而来的压缩程序，文件经它压缩后，会另外产生具有“.lzh”扩展名的压缩文件。

应用实例如下。

压缩一个 lha 压缩文件。

```
#lha godocjh
```

7.14 unarj：解压缩文件

作用：解压缩.arj 文件。

用法：unarj [eltx] [.arj 压缩文件]

主要选项如下。

- e：解压缩.arj 文件。
- l：显示压缩文件内所包含的文件。
- t：检查压缩文件是否正确。
- x：解压缩时保留原有的路径。

说明

unarj 为.arj 压缩文件的压缩程序。

应用实例如下。

解压缩一个.arj 文件。

```
#unarj goodcjh.arj
```

7.15 zip：压缩文件

作用：zip 命令用来压缩文件。

用法：zip [选项] [-b <工作目录>] [-ll] [-n <字尾字符串>] [-t <日期时间>] [-<压缩效率>] [压缩文件] [文件...] [-i <范本样式>] [-x <范本样式>]

主要选项如下。

- -A：调整可执行的自动解压缩文件。
- -b<工作目录>：指定暂时存放文件的目录。
- -c：替每个被压缩的文件加上注释。
- -d：从压缩文件内删除指定的文件。
- -D：压缩文件内不建立目录名称。
- -f：此参数的效果和指定“-u”参数类似，但不仅更新既有文件，如果某些文件原本不存在于压缩文件内，使用本参数会一并将其加入压缩文件中。
- -F：尝试修复已损坏的压缩文件。
- -g：将文件压缩后附加在既有的压缩文件之后，而非另行建立新的压缩文件。
- -h：在线帮助。
- -i<范本样式>：只压缩符合条件的文件。
- -j：只保存文件名称及其内容，而不存放任何目录名称。
- -J：删除压缩文件前面不必要的数据。
- -k：使用 MS-DOS 兼容用法的文件名称。
- -l：压缩文件时，把 LF 字符置换成 LF+CR 字符。
- -ll：压缩文件时，把 LF+CR 字符置换成 LF 字符。
- -L：显示版权信息。
- -m：将文件压缩并加入压缩文件后，删除原始文件，即把文件移到压缩文件中。
- -n<字尾字符串>：不压缩具有特定字尾字符串的文件。
- -o：以压缩文件内拥有最新更改时间的文件为准，将压缩文件的更改时间设成和该文件相同。
- -q：不显示指令执行过程。
- -r：递归处理，将指定目录下的所有文件和子目录一并处理。
- -S：包含系统和隐藏文件。
- -t<日期时间>：把压缩文件的日期设成指定的日期。
- -T：检查备份文件内的每个文件是否正确无误。
- -u：更换较新的文件到压缩文件内。
- -v：显示指令执行过程或显示版本信息。
- -V：保存 VMS 操作系统的文件属性。
- -w：在文件名称里加入版本编号，本参数仅在 VMS 操作系统下有效。
- -x<范本样式>：压缩时排除符合条件的文件。

- -X：不保存额外的文件属性。
- -y：直接保存符号链接，而非该链接所指向的文件，本参数仅在 UNIX 之类的系统下有效。
- -z：替压缩文件加上注释。
- -$：保存第一个被压缩文件所在磁盘的卷册名称。
- -<压缩效率>：压缩效率是一个介于 1～9 之间的数值。

说明

zip 是一个使用广泛的压缩程序，文件经它压缩后会另外产生具有“.zip”扩展名的压缩文件。

应用实例如下。

压缩目录。

```
#zip -r cp3.zip /var/www/
```

7.16 zipinfo：显示压缩文件的信息

作用：zipinfo 命令用来列出压缩文件信息。

用法：zipinfo [选项] [压缩文件] [文件...] [-x <范本样式>]

主要选项如下。

- -1：只列出文件名称。
- -2：此参数的效果和指定“-1”参数类似，但可搭配“-h”、“-t”和“-z”参数使用。
- -h：只列出压缩文件的文件名称。
- -l：此参数的效果和指定“-m”参数类似，但会列出原始文件的大小而非每个文件的压缩率。
- -m：此参数的效果和指定“-s”参数类似，但会列出每个文件的压缩率。
- -M：若信息内容超过一个画面，则采用类似 more 指令的方式列出信息。
- -s：用类似执行“ls -l”指令的效果列出压缩文件内容。
- -t：只列出压缩文件内所包含的文件数目、压缩前后的文件大小及压缩率。
- -T：将压缩文件内每个文件的日期时间用年、月、日、时、分、秒的顺序列出。
- -v：详细显示压缩文件内每一个文件的信息。
- -x<范本样式>：不列出符合条件的文件的信息。
- -z：如果压缩文件内含有注释，就将注释显示出来。

说明

执行 zipinfo 指令可得知.zip 压缩文件的详细信息。

应用实例如下。

（1）以中长格式列出档案中一个单独文件的信息，只需明确指定文件名就可以了。

```
# zipinfo -m storage unshrink.c
```

指定任何一个档案成员的文件名，就像本例一样，将不再显示默认的头末信息；只有显示特定文件信息的那行将被输出。当要获得某个单独文件的信息时，这种模式符合直觉。对于多个文件的情形，知道总的压缩前后的文件大小是很有用的。

（2）为了获得一个 ZIP 档案的完全信息，可使用冗余选项。如果操作系统允许的话，把输出重定向至一个过滤器（如 UNIX 中的 more）通常是明智之举。

```
#zipinfo -v storage | more
```

08 Linux 系统管理命令

8.1 apmd：高级电源管理

作用：apmd 命令负责 BIOS 进阶电源管理（APM）相关的记录、警告与管理工作。

用法：apmd [选项] [-p <百分比变化量>] [-w <百分比值>]

主要选项如下。

- -p<百分比变化量>或--percentage<百分比变化量>：当电力变化的幅度超出设置的百分比变化量时，即记录事件。百分比变化量的预设值为 5；若设置值超过 100，则关闭此功能。
- -u 或--utc：将 BIOS 时钟设为 UTC，以便从悬待模式恢复时，将-u 参数传送至 clock 或 hwclock 程序。
- -v 或--verbose：记录所有的 APM 事件。
- -w<百分比值>或--warn<百分比值>：如果电池不在充电状态，且电池电量低于设置的百分比值，则在 syslog 的 ALERT 层记录警告信息。百分比值的预设值为 10；若设置值为 0，则关闭此功能。
- -W 或--wall：发出警告信息给所有人。

应用实例如下。

（1）当电池的电量低于 15%时，警告所有的用户。

```
# apmd -w 15 -W
```

（2）取消电量低于某个百分比就发出警告的功能。

```
# apmd -q
```

（3）当电量变化幅度超出 15%时，记录事件。

```
# apmd -p 15
```

8.2　apmsleep：APM 进入休眠状态

作用：计算机特别是笔记本电脑可以使用 RTC 设置警报时间，并且进入休眠或备用方式。

用法：apmsleep [选项] [+]hh:mm

主要选项如下。

- -s, --suspend：进入休眠状态。
- -S, --standby：进入备用状态。
- -w, --wait：等待时间。
- -p, --precise：等待警告时间与实际时间匹配。
- -n, --noapm：不使用 APM BIOS 休眠计算机，只设置报警时间和等待时间。
- -d, --debug：在屏幕上输出任务信息。
- hh:mm：指定进入挂起或等待状态的时间。

应用实例如下。

（1）立刻休眠。

```
#apmsleep +0:00
```

（2）进入休眠时间 3 小时。

```
#apmsleep –s +3：00
```

（3）进入备用时间直到第二天 11：00。

```
#apmsleep –S 11：00
```

8.3　apropos：查找使用手册的名字和相关描述

作用：apropos 命令在一些特定的包含系统命令的简短描述数据库文件里查找关键字，然后把结果送到标准输出。

用法：apropos keyword

keyword 即关键字。

主要选项如下。

- -d,--debug：输出调试信息。
- -v,--verbose：输出详细的警告信息。
- -e,--exact：对每个关键词都进行严格匹配的搜索。
- -r,--regex：把每个关键词都当作正则表达式解读。
- -w,--wildcard：关键词里包含通配符。
- -a,--and：要求所有的关键词都同时匹配。
- -l,--long：不要把输出按终端宽度截断。

- -C,--config-file=文件：使用该用户设置文件。
- -L,--locale=区域：定义本次搜索所使用的区域设置。
- -M,--manpath=路径：设置搜索手册页的路径为“路径”。
- -s,--sections=列表,--section=列表：只查找指定的手册 section。

默认开启--regex 选项。

应用实例如下。

（1）查找标题中包含字 useradd 的手册节。

```
# apropos useradd
luseradd    (1)- Add an user
useradd     (8)- create a new user or update default new user information
useradd     (8)- 账 号 建 立 或 更 新 新 使 用 者 的 资 讯
```

（2）查找标题中包含字 editor 的手册节。

```
# apropos editor
alacarte      (rpm) - Simple menu editor for GNOME
ed            (1)- text editor
ex            (1p)- text editor
gedit         (1)- text editor for the GNOME Desktop
gedit         (rpm) - gEdit is a small but powerful text editor for GNOME
nano          (1)- Nano's ANOther editor, an enhanced free Pico clone
ospam         (1)- An SGML/XML markup stream editor
psed          (1)- a stream editor
psed [s2p]    (1)- a stream editor
red [ed]      (1)- text editor
rnano         (1)- Restricted mode for Nano's ANOther editor, an enhanced
free Pico clone
sed           (1)- stream editor for filtering and transforming text
sed           (1p)- stream editor
vi            (1p)- screen-oriented (visual) display editor
vim           (1)- Vi IMproved, a programmers text editor
```

8.4 arch：输出主机的体系结构

作用：显示计算机的体系结构。在目前的 Linux 系统上，arch 显示的数据有“i386”、“i486”、“i586”、“alpha”、“sparc”、“arm”、“m68k”、“mips”、“ppc”，arch 等同于 uname -m 命令。

用法：arch

应用实例如下。

输出当前主机的体系结构。

```
# arch
i686
```

8.5 alias：设置别名

作用：设置命令的别名。

用法：alias[别名]=[命令名称]

> **说明**
>
> 用户可利用 alias 自定义命令的别名。若仅输入 alias，则可列出目前所有的别名设置。alias 的效力仅限于该次登录的操作。若要每次登录时即自动设好别名，可在.profile 或.cshrc 中设定命令的别名。

主要选项如下。

-p：打印列表。

应用实例如下。

（1）更改 ls 命令，使其在列中显示信息并对输出进行注释。

```
#alias ls='ls –CF
```

（2）创建用来重复命令历史文件中的先前条目的命令。

```
#alias r='fc -s'
```

（3）为 du 命令使用 1KB 单元。

```
alias du=du\ -k
```

（4）为用户 Dee 创建一个命令来显示所有活动的进程。

```
alias psc='ps -ef | grep Dee'
```

8.6 cd：切换目录

作用：cd 命令可让用户在不同的目录间切换，但该用户必须拥有足够的权限进入目的目录。

用法：cd [-L|[-P [-e]]] [目录]

主要选项如下。

- -p：如果要切换到的目标目录是一个符号链接，则直接切换到符号链接指向的目标目录。
- -L：如果要切换到的目标目录是一个符号链接，则直接切换到符号链接名代表的目录，而非符号链接所指向的目标目录。
- -：当仅使用“-”一个选项时，当前工作目录将被切换到环境变量“OLDPWD”所表示的目录。

应用实例如下。

进入/home/cjh 目录。

```
#cd /home/cjh
```

8.7 clear：清空终端屏幕

作用：clear 命令用来清空终端屏幕。

用法：clear

应用实例如下。

清空终端屏幕。

```
#clear
```

8.8 clock：系统 RTC 时间设置

作用：调整 RTC 时间。RTC 是计算机内建的硬件时间，执行这项命令可以显示当前时间，调整硬件时钟的时间，将系统时间设为与硬件时钟的时间一致，或把系统时间回存到硬件时钟。

用法：clock [功能] [选项...]

功能如下。

- -r,--show：读取硬件时钟并打印结果。
- --set：将 RTC 设置为--date 指定的时间。
- -s,--hctosys：从硬件时钟设置系统时间。
- -w,--systohc：从当前系统时间设置硬件时钟。
- --systz：基于当前时区设置系统时间。
- --adjust：根据自上次时钟设置或调整后的系统漂移来调整 RTC
- -c,--compare：定期将系统时钟与 CMOS 时钟相比较。
- --getepoch：打印内核的硬件时钟纪元（epoch）值。
- --setepoch：将内核的硬件时钟纪元（epoch）值设置为--epoch 选项指定的值。
- --predict：预测--date 选项所指定时刻读取到的 RTC 值。

- -V,--version：显示版本信息并退出。

主要选项如下。

- -u,--utc：硬件时钟保持为 UTC 时间。
- --localtime：硬件时钟保持为本地时间。
- -f,--rtc<文件>：代替默认文件的特殊/dev/...文件。
- --directisa：直接访问 ISA 总线，而非/dev/rtc。
- --badyear：忽略 RTC 年份（由于 BIOS 损坏）。
- --date<时间>：指定要设置的硬件时钟时间。
- --epoch<年>：指定作为硬件纪元（epoch）值的起始年份。
- --noadjfile：不访问/etc/adjtime；需要使用--utc 或--localtime 选项。
- --adjfile<文件>：指定调整文件的路径；默认为/etc/adjtime。
- --test：不更新，只显示将进行什么操作。
- -D,--debug：调试模式。

应用实例如下。

显示 UTC 时间。

```
[root@-p ~]#clock -u
2014 年 11 月 22 日 星期六 03 时 51 分 01 秒-0.769266 秒
```

8.9 cal：显示日历

作用：显示日历。

用法：cal [选项] [[[日]月]年]

主要选项如下。

- -1,--one：只显示当前月份（默认）。
- -3,--three：显示上个月、当月和下个月。
- -s,--sunday：周日作为一周第一天。
- -m,--monday：周一作为一周第一天。
- -j,--julian：输出儒略日。
- -y,--year：输出整年。

应用实例如下。

（1）显示当前日历。

```
# cal
 三月 2010
日  一  二  三  四  五  六
    1   2   3   4   5   6
 7  8   9   10  11 12  13
```

```
14 15  16  17  18  19 20
21  22  23  24 25  26 27
28  29  30  31
```

（2）显示上个月、当前月、下个月的月历。

```
# cal -3
     二月 2010                 三月 2010                 四月 2010
日 一 二 三 四 五 六      日 一 二 三 四 五 六       日 一 二 三 四 五 六
    1  2  3  4  5  6          1  2  3  4  5  6                    1  2  3
 7  8  9 10 11 12 13       7  8  9 10 11 12 13        4  5  6  7  8  9 10
14 15 16 17 18 19 20      14 15 16 17 18 19 20       11 12 13 14 15 16 17
21 22 23 24 25 26 27      21 22 23 24 25 26 27       18 19 20 21 22 23 24
28                        28 29 30 31                25 26 27 28 29 30
```

（3）显示 2009 年全年日历。

```
# cal -y 2009
```

8.10 chroot：改变根目录

作用：chroot（change root）命令用来把根目录换成指定的目的目录。

用法：chroot [选项] 新根 [命令[参数]...]

　　　chroot 选项

以指定的新根作为运行指定命令时的根目录。

主要选项如下。

- --userspec=用户:组：指定所用的用户及用户组（可使用“数字”或“名字”）。
- --groups=组列表：指定可供选择的用户组列表，形如组 1,组 2,组 3……

应用实例如下。

（1）运行 pwd 命令将/usr/bin 目录作为 root 文件系统。

```
#mkdir /usr/bin/lib
 #cp /usr/ccs/lib/libc.a /usr/bin/lib
 #chroot /usr/bin pwd
```

（2）运行 Korn Shell 子 Shell 将另一个系统文件作为 root 文件系统。

```
#chroot /var/tmp /usr/bin/ksh
```

8.11 date：显示或设置系统时间

作用：date 命令用来显示或设置系统时间与日期。

用法：date [必要选项] [可选选项]

主要选项如下。

- -d,--date=STRING：显示由 STRING 指定的时间，而不是当前时间。
- -f,--file=DATEFILE：显示 DATEFILE 中每一行指定的时间，如同将 DATEFILE 中的每行作为--date 的参数一样。
- -I,--iso-8601[=TIMESPEC]：按照 ISO-8601 的日期/时间格式输出时间。
- TIMESPEC=`date'：（或者不指定时）仅输出日期，等于`hours'、`minutes'或`seconds'时按照指定精度输出日期及时间。
- -r,--reference=FILE：显示 FILE 的最后修改时间。
- -R,--rfc-822：根据 RFC-822 指定格式输出日期。
- -s,--set=STRING：根据 STRING 设置时间。
- -u,--utc,--universal：显示或设置全球时间（格林尼治时间）。

格式：FORMAT 控制着输出格式。仅当选项指定为全球时间时本格式才有效。

- %%：文本的%。
- %a：当前区域的星期几的简写（Sun..Sat）。
- %A：当前区域的星期几的全称（不同长度）（Sunday..Saturday）。
- %b：当前区域的月份的简写（Jan..Dec）。
- %B：当前区域的月份的全称（变长）（January..December）。
- %c：当前区域的日期和时间（SatNov0412:02:33EST1989）。
- %d：（月份中的）几号（用两位表示）（01..31）。
- %D：日期（按照“月/日期/年”格式显示）（mm/dd/yy）。
- %e：（月份中的）几号（去零表示）（1..31）。
- %h：同%b。
- %H：小时（按 24 小时制显示，用两位表示）（00..23）。
- %I：小时（按 12 小时制显示，用两位表示）（01..12）。
- %j：（一年中的）第几天（用 3 位表示）（001..366）。
- %k：小时（按 24 小时制显示，去零显示）（0..23）。
- %l：小时（按 12 小时制显示，去零表示）（1..12）。
- %m：月份（用两位表示）（01..12）。
- %M：分钟数（用两位表示）（00..59）。
- %n：换行。
- %p：当前时间是上午（AM）还是下午（PM）。
- %r：时间，按 12 小时制显示（hh:mm:ss[A/P]M）。
- %s：从 1970 年 1 月 1 日 0 点 0 分 0 秒到现在历经的秒数（GNU 扩充）。

- %S：秒数（用两位表示）（00..60）。
- %：t 水平方向的 tab 制表符。
- %T：时间，按 24 小时制显示（hh:mm:ss）。
- %U：（一年中的）第几个星期，以星期天作为一周的开始（用两位表示）（00..53）。
- %V：（一年中的）第几个星期，以星期一作为一周的开始（用两位表示）（01..52）。
- %w：用数字表示星期几（0..6）；0 代表星期天。
- %W：（一年中的）第几个星期，以星期一作为一周的开始（用两位表示）（00..53）。
- %x：按照（mm/dd/yy）格式显示当前日期。
- %X：按照（%H:%M:%S）格式显示当前时间。
- %y：年的后两位数字（00..99）。
- %Y：年（用 4 位表示）（1970...）。
- %z：按照 RFC-822 中指定的数字时区显示（如，-0500）（为非标准扩充）。
- %Z：时区。例如，EDT（美国东部时区）如果不能决定是哪个时区则为空。

默认情况下，用 0 填充数据的空缺部分。GNU 的 date 命令能分辨在`%'和数字指示之间的以下修改。

`-' (连接号)，不进行填充；`_'(下画线)，用空格进行填充。若不加任何参数，data 会显示目前的日期与时间。

应用实例如下。

（1）显示当前时间。

```
# date
2008 年 11 月 14 日星期五 07:41:29 CST
```

（2）按照指定的格式显示日期和时间。

```
#date +"%r %a %d %h %y
```

（3）设置日期和时间。

```
date 0217142590
```

对于使用 CST 时区的系统，这项输入设置日期和时间为 CST 1990 年 2 月 17 日 14:25:00 星期六。注意，必须拥有 root 用户权限才能更改日期和时间。

8.12 dmesg：显示开机信息

作用：dmesg 命令用来显示开机信息。Kernel 会将开机信息存储在 ring buffer 中。如果开机时来不及查看信息，可利用 dmesg 命令来查看。开机信息也保存在/var/log 目录中名为 dmesg 的文件里。

用法：dmesg [选项]

主要选项如下。

- -C,--clear：清除内核环形缓冲区（Ring Butter）。
- -c,--read-clear：读取并清除所有消息。
- -D,--console-off：禁止向终端打印消息。
- -d,--show-delta：显示打印消息之间的时间差。
- -e,--reltime：以易读格式显示本地时间和时间差。
- -E,--console-on：启用向终端打印消息。
- -F,--file<文件>：用文件代替内核日志缓冲区。
- -f,--facility<列表>：将输出限制为定义的设施。
- -H,--human：易读格式输出。
- -k,--kernel：显示内核消息。
- -L,--color：显示彩色消息。
- -l,--level<列表>：限制输出级别。
- -n,--console-level<级别>：设置打印到终端的消息级别。
- -P,--nopager：不将输出通过管道传递给分页程序。
- -r,--raw：打印原生消息缓冲区。
- -S,--syslog：强制使用 syslog(2)而非/dev/kmsg。
- -s,--buffer-size<大小>：查询内核环形缓冲区所用的缓冲区大小。
- -T,--ctime：显示易读的时间戳（如果用户使用了 SUSPEND/RESUME 则可能不准）。
- -t,--notime：不打印消息时间戳。
- -u,--userspace：显示用户空间消息。
- -w,--follow：等待新消息。
- -x,--decode：将设施和级别解码为可读的字符串。

支持的日志设施如下。

- kern：内核消息。
- user：随机的用户级消息。
- mail：邮件系统。
- daemon：系统守护进程。
- auth：安全/认证消息。
- syslog：syslogd 内部生成的消息。
- lpr：行打印机子系统。
- news：网络新闻子系统。

支持的日志级别（优先级）如下。

- emerg：系统无法使用。
- alert：操作必须立即执行。
- crit：紧急条件。
- err：错误条件。
- warn：警告条件。

– notice：正常但重要的条件。
– info：信息。
– debug：调试级别的消息。

应用实例如下。

显示开机信息。

```
# dmesg
```

8.13 dircolors：设置 ls 命令在显示目录或文件时所用的色彩

作用：dircolors 可根据色彩配置文件来设置 LS_COLORS 环境变量或显示设置 LS_COLORS 环境变量的 Shell 命令。

用法：dircolors[色彩配置文件]

dircolors [-bcp] [--help] [--version]

主要选项如下。

– -b 或--sh 或--bourne -shell：显示在 Boume Shell 中，将 LS_COLORS 设为目前预设置的 Shell 命令。
– -c 或--csh 或--c -shell：显示在 C Shell 中，将 LS_COLORS 设为目前预设置的 Shell 命令。
– -p 或--print -database：显示预设置。

应用实例如下。

显示预设置。

```
# dircolors -p
```

8.14 depmod：分析模块

作用：depmod（depend module）分析可载入模块的相依性，供 modprobe 在安装模块时使用。

用法：depmod [-adeisvV] [-m <文件>] [--help] [模块名称]

主要选项如下。

– -a 或--all：分析所有可用的模块。
– -d 或 debug：执行排错模式。
– -e：输出无法参照的符号。
– -i：不检查符号表的版本。
– -m<文件>或 system-map<文件>：使用指定的符号表文件。
– -s 或--system -log：在系统记录中记录错误。

– -v 或--verbose：执行时显示详细的信息。
– -b basedir 或--basedir basedir：若用户的模块并没有正确地在/lib/mdules/version 下，可以指定目录生成依赖。
– -F,--filesyms=FILE：提供一个 System.map 文件（在内核编译时生成的）的内核符号。
– -A –quick：检查是否有模块比 modues.dep 中的模块新，若没有，则退出而不重新生成文件。

应用实例如下。

分析所有可用的模块。

```
# depmod –a
```

8.15 echo：显示文本行

作用：echo 命令用来显示一行文本。

作用：echo[选项]... [STRING]...

主要选项如下。

– -n：不输出行尾的换行符。
– -e：允许对下面列出的加反斜线转义的字符进行解释。
– -E：禁止对在 STRINGs 中的那些序列进行解释。在没有-E 选项的情况下，可承认并可以内置替换以下序列。
  Ø \NNN：字符的 ASCII 代码为 NNN（八进制）。
  Ø \\：反斜线。
  Ø \a：报警符（BEL）。
  Ø \b：退格符。
  Ø \c：禁止尾随的换行符。
  Ø \f：换页符。
  Ø \n：换行符。
  Ø \r：回车符。
  Ø \t：水平制表符。
  Ø \v：纵向制表符。
– --help：显示帮助并退出（需单独运行）。
– --version：输出版本信息并退出（需单独运行）。

应用实例如下。

（1）将一个消息写到标准输出。

```
echo Please insert diskette ...
```

（2）将报文写到标准错误输出。

```
echo Error: file already exists. >&2
```

这个命令将错误信息重定向到标准错误。如果>&2 被省略了，那么报文将被写到标准输出中。

8.16 exec：执行完命令后交出控制权

作用：exec 用于执行完命令后交出控制权。

用法：exec[-cl] [-a 名称] [命令[参数...]] [重定向...]

主要选项如下。

-c：在空环境中执行指定的命令。

应用实例如下。

执行 ls 命令后退出。

```
#exec ls
```

8.17 exit：退出 Shell

作用：exit 命令用于退出目前的 Shell。执行 exit 可使 Shell 以指定的状态值退出。若不设置状态值参数，则 Shell 以预设值退出。状态值 0 代表执行成功，其他值代表执行失败。exit 也可用于 script，离开正在执行的 script，回到 Shell。

用法：exit

应用实例如下。

退出 Shell。

```
#exit
```

8.18 eject：弹出介质

作用：eject 命令用来退出抽取式设备。若设备已挂入，则 eject 会先将该设备卸除再退出。

用法：eject [选项] [<设备>|<挂载点>]

主要选项如下。

- -a,--auto<on|off>：打开/关闭自动弹出功能。
- -c,--changerslot<slot>：在 CD-ROM 更换器上换盘。

- -d,--default：显示默认设备。
- -f,--floppy：弹出软盘。
- -F,--force：无视设备类型。
- -i,--manualeject<on|off>：切换手动弹出保护开/关。
- -m,--no-unmount：不卸载设备，即使它已挂载。
- -M,--no-partitions-unmount：不卸载另外的分区。
- -n,--noop：不弹出，只显示找到的设备。
- -p,--proc：使用/proc/mounts 代替/etc/mtab。
- -q,--tape：弹出磁带。
- -r,--cdrom：弹出 CD-ROM。
- -s,--scsi：弹出 SCSI 设备。
- -t,--trayclose：关闭托盘。
- -T,--traytoggle：开/关托盘。
- -v,--verbose：启用详尽输出。
- -x,--cdspeed<speed>：设置 CD-ROM 最高速度。
- -X,--listspeed：列出 CD-ROM 可用速度。

默认将依次尝试 -r、-s、-f 和 -q 直到成功。

应用实例如下。

（1）弹出光驱。

```
#eject /dev/cdrom
```

（2）弹出指定设备。

```
# eject -r /dev/sda4
```

说明

eject 允许可移动介质（典型的是 CD-ROM、软盘、磁带或者 JAZ 及 ZIP 磁盘）在软件控制下弹出。该命令也可以控制一些多盘片 CD-ROM 控制器，一些设备支持的自动弹出功能，以及一些 CD-ROM 驱动器磁盘托盘的关闭。与 name 相对应的设备将被弹出。name 可以是设备文件或者其挂载点，也可以是完整路径或者省略前面的/dev 或者/mnt 设备文件名。如果没有指定 name，默认使用 cdrom。eject 只对支持 4 种弹出方式其中之一或更多的设备起作用。这些设备包括大多数的 CD-ROM 驱动器（IDE、SCSI 和专有接口的）、一些 SCSI 磁带驱动器、JAZ 驱动器、ZIP 驱动器（并口，SCSI 及 IDE 版本接口的），以及 LS120 可移动软盘。如果 eject 判断设备能有多个分区，它会尝试在弹出前卸载所有该设备已挂载的分区。如果卸载失败，程序不会尝试弹出该介质。

8.19 enable：启动或关闭 Shell 命令

作用：enable 命令用于启动或关闭 Shell 内建命令。

用法：enable [选项] [内建命令]

主要选项如下。

- -n：关闭指定的 Shell 内建命令。
- -a：显示 Shell 所有关闭与启动的命令。
- -a：显示所有激活的内部命令。
- -f：从指定文件中读取内部命令。

> **说明**
>
> 若要执行的文件名称与 Shell 内建命令相同，可用 enable -n 来关闭 Shell 内建命令。若不加-n 参数，enable 可重新启动关闭的命令。

应用实例如下。

（1）关闭 fg 命令。

```
# enable fg
```

（2）重新启动关闭的命令。

```
# enable
```

8.20 fc：修改或执行命令

作用：fc 命令用来修改或执行命令。

用法：fc [-e ename] [-nlr] [first] [last]

fc -s [pat=rep] [cmd]

主要选项如下。

- -e<文本编辑程序>：指定使用的文本编辑程序，默认编辑器顺序是 FCEDIT、EDITOR、vi。
- -l：仅列出首个和末尾命令范围内的所有命令，而不是编辑。
- -n：显示命令列表时不显示编号。
- -r：显示命令列表时采用逆向排序。
- -s<命令日志内的命令>：在命令日志内由后往前查找符合条件的最后一条命令并执行。

应用实例如下。

列出之前运行过的 5 个命令，而不是对它们进行编辑。

```
#fc -l -5
523    enable
524    fc -l -5
525    ls
526    cat
527    cal
```

8.21 fgconsole：显示虚拟终端的数目

作用：fgconsole 命令用来显示当前虚拟终端的数目。

用法：fgconsole

主要选项如下。

-n --next-available：显示下一个未被分配的 VT 编号。

应用实例如下。

显示当前虚拟终端的数目。

```
#fgconsole
1
```

8.22 free：显示内存信息

作用：free 命令用来显示内存状态，包括实体内存、虚拟的交换文件内存、共享内存区段，以及系统核心使用的缓冲区等。

用法：free [选项]

主要选项如下。

- -b：以 Byte 为单位显示内存使用情况。
- -k：以 KB 为单位显示内存使用情况。
- -m：以 MB 为单位显示内存使用情况。
- -o：不显示缓冲区调节列。
- -s<间隔秒数>：持续观察内存使用状况。
- -t：显示内存总和列。
- -V：显示版本信息。

应用实例如下。

以总和形式显示内存使用情况。

```
# free -t
             total       used       free     shared    buffers     cached
Mem:         299704      294912     4792     0         37448       174724
-/+ buffers/cache:       82740      216964
Swap:        524280      0          524280
Total:       823984      294912     529072
```

8.23 fwhois：显示用户的信息

作用：fwhois 命令的功能有点类似于 finger 命令，可以查找并显示指定账号的用户相关信息。不同之处在于 fwhois 命令是到 Network Solutions 的 WHOIS 数据库中去查找，该账号名称必须在上面注册才能寻获，且名称没有大小写的差别。

用法：fwhois [username]

8.24 getty：设置终端模式

作用：getty（get teletypewriter）命令用来设置终端模式、连线速率和管制线路。

用法：getty [-h] [-d<组态配置文件>] [-r<延迟秒数>] [-t<超时秒数>] [-w<等待字符串>] [终端机编号] [连线速率<终端机类型><管制线路>]

getty [-c<定义配置文件>]

主要选项如下。

- -c<定义配置文件>：指定定义配置文件，预设为/etc/gettydefs。
- -d<组态配置文件>：指定组态配置文件，预设为/etc/conf.getty。
- -h：当传输速率为 0 时强制断线。
- -r<延迟秒数>：设置延迟时间。
- -t<超时秒数>：设置等待登录的时间。
- -w<等待字符串>：设置等待回应的字符串。

说明

getty 命令是 UNIX 类操作系统启动时所必需的 3 个步骤之一。

应用实例：

开启终端 7。

```
# getty tty7
```

8.25 gitps：显示程序情况

作用：gitps 是用来报告并管理程序执行的指令，基本上它就是通过 ps 指令来报告、管理程序的，也能通过 gitps 指令随时中断、删除不必要的程序。因为 gitps 指令会去执行 ps 指令，所以其参数和 ps 指令类似。

用法：gitps [选项] [p <进程号>] [t <终端>] [U <用户>]

主要选项如下。

- -a：显示所有进程的信息。
- -c：仅显示进程的真实名。
- -e：显示环境变量。
- -f：显示进程间的关系。
- -n：用数字显示。
- -p <进程号>：指定进程。
- -t <终端>：指定终端。
- -U <用户>：指定用户。
- -v：以虚拟内存的形式显示。
- -x：不区分终端。

应用实例如下。

（1）显示指定用户的进程信息。例如，显示用户 root 的进程信息。

```
#gitps root
```

（2）显示所有进程。

```
#gitps –a
```

8.26 logwatch：可定制和可插入式的日志监视系统

作用：logwatch 命令是一个可定制和可插入式的日志监视系统，它通过遍历给定时间范围内的系统日志文件而产生日志报告。logwatch 默认每天执行一次，可以从/etc/cron.daily 里看到。

用法：logwatch 选项

主要选项如下。

- --detail<报告详细程度>：指定日志报告的详细程度。

- --logfile<日志文件>：仅处理指定的日志文件。
- --service<服务名>：仅处理指定服务的日志文件。
- --print：打印结果到标准输出。
- --mailto<邮件地址>：将结果发送到指定邮箱。
- --range<日期范围>：指定处理日志的日期范围。
- --archives：处理归档日志文件。
- --debug<调试等级>：调试模式。
- --save<文件名>：将结果保存到指定文件中，而不显示或者发送到指定邮箱。
- --logdir<目录>：指定查找日志文件的目录，而不使用默认的日志目录。
- --hostname<主机名>：指定在日志报告中使用的主机名，而不使用系统默认的主机名。
- --numeric：在报告中显示 IP 地址而不是主机名。
- --help：显示指令的帮助信息。

应用实例如下。

（1）显示出所有日志的情况。

```
# logwatch --detail High --Service All --range All --print
```

（2）只看 sshd 的日志情况。

```
# logwatch --service sshd --detail High --print
```

8.27 logsave：把一个命令的输出输出到一个指定的日志文件中

作用：logsave 程序执行 cmd_prog（可以具带参数），然后把这个程序的输出复制到一个指定的文件 logfile 中，如果文件不存在，就写在内存中直到可以写为止。写入文件的同时也会把输出输出到标准输出上面。如果 cmd_prog 是“-”，那么 logsave 将会把标准输入的内容保存到一个 log 文件中。

用法：logsave [-asv] logfile program

主要选项如下。

- -a：把内容追加到 logfile 中，而不是替换。
- -s：这个选项导致 logsave 忽略向 log 文件写入这样的字符串，即字符串被 control-A（ASCII 为 001 或者头的开始）和 control-B（ASCII 为 002 或者文本的开始）括起来。这样，像进度条的信息就可以在用户终端中被看见，但是却没有写入到 log 文件中。
- -v：可以显示更多的信息。

应用实例如下。

把 ls 命令输出到文件中。

```
$logsave testOut ls
```

8.28 GRUB：引导加载程序

正确安装 Linux 操作系统后，可从硬盘引导系统进入 GRUB，启动菜单界面如图 8-1 所示。在该界面中可以使用的按键如表 8-1 所示。在该菜单界面中可以选择 GRUB 配置文件。

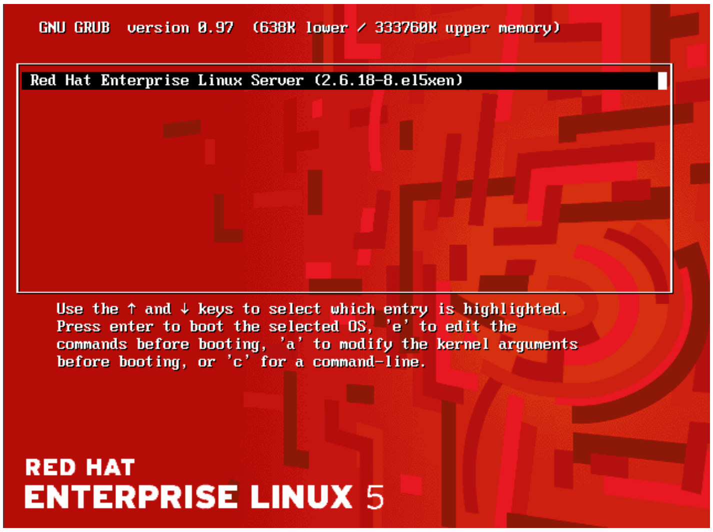


图 8-1　GRUB 启动菜单界面

表 8-1　GRUB 启动菜单按键

按　键	功能说明
↑　↓	使用上、下方向键在启动菜单项之间进行移动
Enter	使用 Enter 键启动当前的菜单项
e	使用 e 键编辑当前的启动菜单项
a	使用 a 键添加内核的启动参数
c	使用 c 键进入 GRUB 的命令行方式

1．GRUB 的启动菜单项编辑界面

在 GRUB 的启动菜单界面中使用“e”键进入 GRUB 的启动菜单项编辑界面，如图 8-2 所示。在该界面中可以使用的操作按键如表 8-2 所示。该界面提供了灵活的配置接口，对于调试操作系统启动配置非常有用。

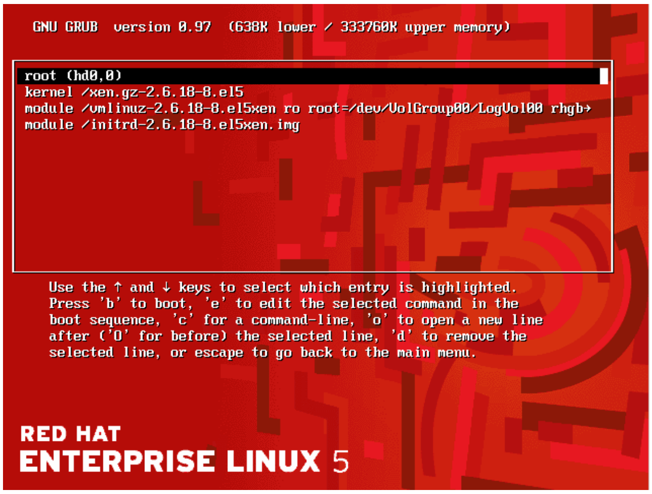


图 8-2　GRUB 的启动菜单项编辑界面

表 8-2　GRUB 启动菜单项编辑界面按键

按　键	功能说明
↑　↓	使用上、下方向键在启动菜单项之间进行移动
b	使用 b 键启动当前的菜单项
e	使用 e 键编辑当前选中的行
d	使用 d 键删除当前行
c	使用 c 键进入 GRUB 的命令行方式
o	使用 o 键在当前行后面插入一行
O	使用 O 键在当前行前面插入一行
Esc	使用 Esc 键返回

2．GRUB 命令行界面

GRUB 有两种方法可以进入命令行界面，即从 GRUB 启动菜单进入或者在 Shell 状态下使用 grub 命令进入，使用两种方法获得的命令行界面稍有不同。由于 GRUB 命令是运行在 Linux 操作系统中的，受操作系统的限制，很多命令不能使用。而从 GRUB 启动菜单进入命令行界面支持的命令比较完整。

1）GRUB 命令行界面的特点

GRUB 命令行界面提供了方便友好的命令行交互方式，其主要特点如下：

- 提供在线帮助命令“help”，并且可以获得每条命令的详细帮助。
- 可使用左、右方向键编辑行命令。
- 可使用上、下方向键滚动历史命令。
- 可使用 Tab 键补全命令和路径。

2）从 GRUB 启动菜单进入命令行界面

从 GRUB 的启动菜单界面或菜单项编辑界面使用 c 键可进入 GRUB 的命令行界面，如图 8-3 所示。

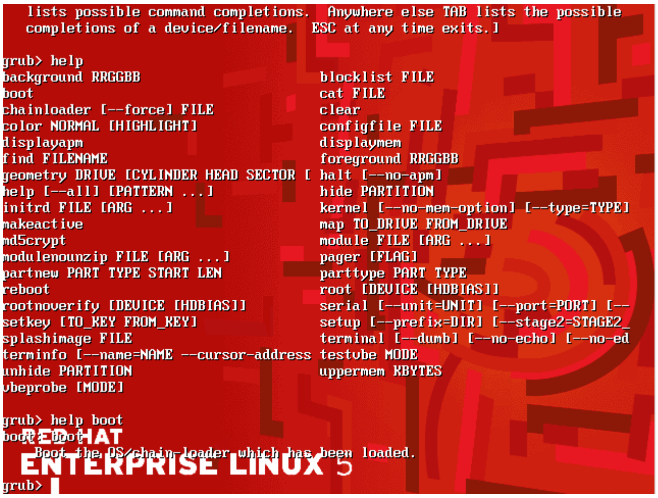


图 8-3　使用 help 命令获得 GRUB 当前可使用的命令

说明

使用 Esc 键可返回菜单界面；使用 help 命令可获得 GRUB 当前可使用的命令；把某个命令作为 help 命令的参数，可获得该命令的详细帮助说明。

3）从 Linux 的 Shell 进入 GRUB 命令行界面

使用 grub 命令也可以进入 GRUB 命令行界面，该命令的完整路径为“/sbin/grub”。

3．GRUB 命令参考

GRUB 中的命令可分为 3 类。

- 菜单命令，只能用于配置文件的全局配置部分。
- 常规命令，既能用于配置文件的全局部分，又能在命令行界面中使用。
- 命令行和菜单项命令，既能用于配置文件的菜单项定义部分，又能用于命令行界面。

1）菜单命令

菜单命令只能用于 GRUB 配置文件的全局配置部分，不能用在 GRUB 命令行交互界面。菜单命令在配置文件中应放在其他命令之前。可使用的菜单命令如表 8-3 所示。

表 8-3　GRUB 菜单命令

菜单命令	功能说明
default	设置默认启动的菜单项
fallback	设置启动某菜单项失败后返回的菜单项
hiddenmenu	隐藏菜单界面
timeout	设置菜单自动启动的延时时间
title	开始一个菜单项

2）常规命令

常规命令可以应用于配置文件和GRUB命令行交互界面。可使用的常规命令如表8-4所示。

表 8-4　常规命令

常规命令	功能说明
bootp	通过 BOOTP 初始化网络设备
color	设置菜单界面的颜色
device	指定设备文件作为驱动器
dhcp	通过 DHCP 初始化网络设备
hide	隐藏某分区
ifconfig	手工配置网络设备
pager	改变内部页程序的状态
partnew	新建一个主分区
parttype	改变分区的类型
password	为菜单界面设置密码
rarp	通过 RARP 初始化网络设备
unhide	还原某隐藏分区
serial	设置串口设备
setkey	设置键盘映射
splashimage	设置 GRUB 启动的背景图形文件
terminal	选择终端类型
tftpserver	指定 TFTP 服务器

3）命令行和菜单项命令

命令行和菜单项命令可应用于 GRUB 配置文件的菜单项设置中，也可以用在 GRUB 命令行交互界面，该类命令如表 8-5 所示。

表 8-5　命令行和菜单项命令

命令行和菜单项命令	功能说明
Cat	显示文件内容
boot	启动操作系统
blocklist	显示某文件所在的分区位置（block list notation）
chainloader	把启动控制权交给另外的启动引导器
Cmp	比较两个文件
configfile	加载已存在的 GRUB 配置文件
debug	设置 GRUB 为 debug 模式
displayapm	显示 APM BIOS 信息
displaymem	显示内存配置

续表

命令行和菜单项命令	功能说明
embed	嵌入 Stage 1.5 文件
Find	查找包括某文件的所有设备
fstest	测试文件系统
geometry	显示某驱动器的物理信息
halt	停止计算机运行（软件关机）
Help	显示 GRUB 的命令帮助信息
impsprobe	查询对称多处理器（SMP）信息
initrd	加载 initrd 文件
install	安装 GRUB
ioprobe	查询某驱动器的输入/输出（I/O）端口
kernel	引导操作系统内核
lock	锁定某 GRUB 引导菜单项，使其输入密码后才可启动
makeactive	激活某主分区
map	虚拟映射某驱动器
md5crypt	使用 MD5 算法加密密码
module	加载模块
modulenounzip	加载模块不进行解压
pause	暂停并等待按键
quit	退出 GRUB
reboot	重新启动计算机
read	读取内存中的内容
root	设置 GRUB 的 root 设备
rootnoverify	设置 GRUB 的 root 设备但不装载文件系统
savedefault	保存当前的启动菜单项为默认启动
setup	自动安装 GRUB
testload	从文件系统中测试读取某文件
testvbe	测试 VESA BIOS EXTENSION
uppermem	强制设置主机上位内存的大小
vbeprobe	查询 VESA BIOS EXTENSION 信息

8.29 halt：关闭系统

作用：halt 命令会先检测系统的 runlevel。若 runlevel 为 0 或 6，则关闭系统；否则即调用 shutdown 来关闭系统。

作用：halt[选项]

主要选项如下。

- -q, --quiet：引起快速停机。
- -d：不要在 wtmp 中记录。
- -f：不论目前的 runlevel 为何，不调用 shutdown 强制关闭系统。
- -i：执行 halt 之前，关闭全部的网络界面。
- -n：执行 halt 之前，不用先执行 sync。
- -p：执行 halt 之后，执行 poweroff。
- -w：仅在 wtmp 中记录，而不实际结束系统。

说明

只有 root 用户可以使用此命令。如果其他用户登录系统，就不能使用此命令。halt 命令使用 syslogd 命令记录关机，且把一个关机记录放在/var/log/wtmp 登录记账文件中。系统也把一个条目写入错误记录里，声明系统已关机。

应用实例如下。

（1）立刻关闭系统。

```
#halt -p
```

（2）模拟关闭系统。

```
#halt -w
```

8.30 history：显示历史命令

作用：history 命令用来显示历史命令列表。

用法：history[选项]文件

主要选项如下。

- -c：清空历史列表，删除所有条目。
- -d offset：删除 offset 位置的历史条目。
- -a：将“新”的历史条目（自当前 bash 会话开始输入的历史命令）追加到历史文件中。
- -n：将尚未从历史文件中读取的历史条目读入当前历史列表。这些行是当前 bash 会话开始之后才追加到历史文件中的行。
- -r：读取历史文件的内容，使用它们作为当前历史。
- -w：将当前历史列表写入历史文件，覆盖历史文件的原有内容。
- -p：对后续的 args 进行历史替换，在标准输出上显示结果。不会将结果存入历史列表。

每个 args 都必须被引用。

- -s：将 args 保存到历史列表中，作为单独的条目。历史列表中的最后一条命令在添加 args 之前被删除。

应用实例如下。

（1）查看历史记录命令。

```
#history
```

查看以前使用过的命令，默认最大历史记录条数为 1000（Red Hat 中），可以在/etc/profile 里面修改 HISTSIZE 参数来修改默认值。

（2）清除历史记录命令。

```
#history –c
```

8.31 hwclock：显示与设定硬件时钟

作用：hwclock（hardware clock）命令用来显示与设定硬件时钟。

用法：hwclock [功能] [选项...]

功能如下。

- -r,--show：读取硬件时钟并打印结果。
- --set：将 RTC 设置为--date 指定的时间。
- -s,--hctosys：从硬件时钟设置系统时间。
- -w,--systohc：从当前系统时间设置硬件时钟。
- --systz：基于当前时区设置系统时间。
- --adjust：根据自上次时钟设置或调整后的系统漂移来调整 RTC。
- -c,--compare：定期将系统时钟与 CMOS 时钟相比较。
- --getepoch：打印内核的硬件时钟纪元（epoch）值。
- --setepoch：将内核的硬件时钟纪元（epoch）值设置为--epoch 选项指定的值。
- --predict：预测--date 选项所指定时刻读取到的 RTC 值。

主要选项如下。

- -u,--utc：硬件时钟保持为 UTC 时间。
- --localtime：硬件时钟保持为本地时间。
- -f,--rtc<文件>：代替默认文件的特殊/dev/...文件。
- --directisa：直接访问 ISA 总线，而非/dev/rtc。
- --badyear：忽略 RTC 年份（由于 BIOS 损坏）。
- --date<时间>：指定要设置的硬件时钟时间。
- --epoch<年>：指定作为硬件纪元（epoch）值的起始年份。
- --noadjfile：不访问/etc/adjtime；需要使用--utc 或--localtime 选项。
- --adjfile<文件>：指定调整文件的路径；默认为/etc/adjtime。

– --test：不更新，只显示将进行什么操作。
– -D,--debug：调试模式。

说明

在 Linux 系统中有硬件时钟与系统时钟两种时钟。硬件时钟是指主板上的时钟设备,也就是通常可在 BIOS 画面设定的时钟。系统时钟则是指 Kernel 中的时钟。当 Linux 启动时，系统时钟会去读取硬件时钟的设定，之后系统时钟即独立运作。所有 Linux 相关命令与函数都读取系统时钟的设定。

应用实例如下。

（1）显示硬件时钟。

```
# hwclock
2008 年 08 月 14 日星期四 09 时 55 分 53 秒-0.472837 seconds
```

（2）设置硬件时间为 2008 年 9 月 22 日星期一 16 时 45 分 05 秒。

```
#hwclock --set --date="9/22/08 16:45:05"
```

8.32 init：进程处理初始化

作用：init 命令用来进行进程处理初始化。

用法：init [选项]

主要选项如下。

– -s,S,single：以单用户模式启动系统。按这种模式启动时，/etc/inittab 需要被检查，并在单用户模式 Shell 启动之前会执行引导 rc 脚本。
– 1-5：定义需要启动的运行级别。
– -b,emergency：不运行任何启动脚本而直接进入单用户模式 Shell。
– -a,auto：如果用默认命令行从内核引导（没有用户干预），LILO 引导管理器会向命令行增加相关参数。
– -z xxx：-z 参数被忽略。可以用这个选项略微扩展命令行，这样它可以在堆栈里获得更多空间。然后 init 就可以操作命令行，好让 ps(1)显示当前运行级别。
– 环境变量：init 为所有的子进程设定下列环境变量。
  Ø PATH：/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin。
  Ø INIT_VERSION：如名字表示的那样，在决定一个脚本是否直接来自 init 时很有用。
  Ø RUNLEVEL：当前系统的运行级别。
  Ø PREVLEVEL：前一次运行的运行级别（仅当改变运行级别时有用）。

- CONSOLE：系统控制台。此变量由内核继承而来。如果没有此环境变量的定义，init 会使用/dev/console 作为默认的控制台。

> **说明**
>
> init 是所有进程的父进程。它的首要任务是从一个存储在文件/etc/inittab 中的脚本里创建进程。这个文件通常有一些令 init 在每条用户可登录的线上派生 getty 的记录。它同时控制着各独特系统所要求的独立进程。

应用实例如下。

（1）关闭系统。

```
#init 0
```

（2）重启系统。

```
#init 6
```

8.33 last：显示登录用户信息

作用：last 命令用来列出目前与过去登录系统的用户相关信息。

作用：last [选项] [-f <记录文件>] [-n <显示列数>] [账号名称...] [终端机编号...]

主要选项如下。

- -t tty：只显示指定的虚拟控制台上的登录情况。
- -h 节点：只显示指定的节点上的登录情况。
- -i IP：只显示指定的 IP 上的登录情况。
- -1：用 IP 来显示远端地址。
- -F：显示记录的年、月、日。
- -ID：显示查询的用户名。
- -a：把从何处登录系统的主机名称或 IP 地址显示在最后一行。
- -d；将 IP 地址转换成主机名称。
- -f<记录文件>：指定记录文件。
- -n<显示列数>或-<显示列数>：设置列出名单的显示列数。
- -R：不显示登录系统的主机名称或 IP 地址。
- -x：显示系统关机、重新开机及执行等级的改变等信息。

应用实例如下。

显示最后登录的 3 个用户信息。

```
# last -3
root    tty1                    Thu Aug 14 10:26   still logged in
```

```
cjh     tty1                         Thu Aug 14 09:39 - 10:21(00:41)
cjh1    pts/1   192.168.40.11        Thu Aug 14 09:39  still logged in
```

8.34 lilo：引导管理器

作用：安装核心载入、开机管理程序。

用法：lilo [-clqtV] [-b<外围设备代号>] [-C<配置文件>] [-d<延迟时间>] [－D<识别标签>] [-f<几何参数文件>] [-i<开机磁区文件>] [-I<识别标签>] [-m<映射文件>] [-P] [-r<根目录>] [-R<执行命令>...] [-s<备份文件>] [-S<备份文件>] [-uU<外围设备代号>] [-v...]

主要选项如下。

- -b<外围设备代号>：指定安装 lilo 处的外围设备代号。
- -c：使用紧致映射模式。
- -C<配置文件>：指定 lilo 的配置文件。
- -d<延迟时间>：设置开机延迟时间。
- -D<识别标签>：指定开机后预设启动的操作系统，或系统核心识别标签。
- -f<几何参数文件>：指定磁盘的几何参数配置文件。
- -i<开机磁区文件>：指定欲使用的开机磁区文件，预设是/boot 目录里的 boot.b 文件。
- -I<识别标签>：显示系统核心存放之处。
- -l：产生线形磁区地址。
- -m<映射文件>：指定映射文件。
- -P：决定要修复或忽略分区表的错误。
- -q：列出映射的系统核心文件。
- -r<根目录>：设置系统启动时欲挂入成为根目录的目录。
- -R<执行命令>：设置下次启动系统时，首先执行的命令。
- -s<备份文件>：指定备份文件。
- -S<备份文件>：强制指定备份文件。
- -t：不执行命令，仅列出实际执行会进行的动作。
- -u<外围设备代号>：删除 lilo。
- -U<外围设备代号>：此参数的效果和指定“-u”参数类似，但不检查时间戳记。
- -v：显示命令执行过程。
- -V：显示版本信息。

说明

lilo 是 Linux 系统的一个核心载入程序，同时具备管理开机的功能。单独执行 lilo 命令，它会读取/etc/目录下的 lilo.conf 配置文件，然后根据其内容安装 lilo。

8.35 login：登录系统

作用：login 命令用来登录系统。

用法：login[-p] [-h 主机] [-H] [-f 用户名|用户名]

主要选项如下。

- -p：告诉 login 指令不销毁环境变量。
- -h：指定远程服务器的主机名。

> **说明**
>
> login 命令让用户登录系统，也可通过它的功能随时更换登录身份。在 Slackware 发行版中，可在命令后面附加欲登录的用户名称，它会直接询问密码，等待用户输入。当/etc 目录里含有名为 nologin 的文件时，系统只允许 root 账号登录系统，其他用户一律不准登录。

应用实例如下。

如果选择用命令行模式登录 Linux 的话，那么看到的第一个 Linux 命令就是 login。

一般界面是这样的：

```
Manddrake Linux release 9.1（Bamboo） for i586
renrel 2.4.21-0.13mdk on i686 / tty1
localhost login:root
password:
```

上面代码中，第一行是 Linux 发行版本号，第二行是内核版本号和登录的虚拟控制台，我们在第三行输入登录名，按 Enter 键在 Password 后输入账户密码，即可登录系统。出于安全考虑，输入账户密码时字符不会在屏幕上回显，光标也不移动。

登录后会看到下面这个界面（以超级用户为例）：

```
[root@localhost root]#
last login:Tue ,Nov 18 10:00:55 on vc/1
```

上面显示的是登录星期、月、日、时间和使用的虚拟控制台。

8.36 local：显示本地支持的语言系统信息

作用：local 命令用来显示本地支持的语言系统信息。

用法：locale [选项...] NAME

　　　locale [选项...] [-a|-m]

主要选项如下。

1．系统信息

- -a, --all-locales：写出可用区域的名称。
- -m, --charmaps：写出可用字符映射的名称。

2．修改输出作用

- -c, --category-name：写出选中范围的名称。
- -k, --keyword-name：写出选中关键字的名称。
- -v, --verbose：打印更多信息。
- -?, --help：给出该系统求助列表。
- --usage：给出简要的用法信息。
- -V, --version：打印程序版本号。

说明

用户可以通过设置语系相关的环境变量修改当前的语言系统。超级用户也可以通过修改配置文件/etc/sysconfig/i18n 改变语言系统。

应用实例如下。

查看当前语言系统，使用如下命令：

```
#locale
LANG=zh_CN.UTF-8
LC_CTYPE="zh_CN.UTF-8"
LC_NUMERIC="zh_CN.UTF-8"
LC_TIME="zh_CN.UTF-8"
LC_COLLATE="zh_CN.UTF-8"
LC_MONETARY="zh_CN.UTF-8"
LC_MESSAGES="zh_CN.UTF-8"
LC_PAPER="zh_CN.UTF-8"
LC_NAME="zh_CN.UTF-8"
LC_ADDRESS="zh_CN.UTF-8"
LC_TELEPHONE="zh_CN.UTF-8"
LC_MEASUREMENT="zh_CN.UTF-8"
LC_IDENTIFICATION="zh_CN.UTF-8"
LC_ALL=
```

8.37 logout：退出系统

作用：logout 命令用来退出登录 Shell，其功能和 login 命令对应。

用法：logout

应用实例如下。

退出系统。

```
#logout
```

8.38 logrotate：处理 Log 文件

作用：logrotate 命令用来处理 Log 文件。

用法：logrotate [-dv] [-f | --force] [-s | --state file] config_file+

主要选项如下。

- -d：开启 debug 模式，并且默认包含选项-v，没有任何记录文件或者状态文件会被修改或变动。
- -f,--force：强制执行文件维护操作，即使 logrotate 命令认为该操作是不必要的。
- -m,--mail <command>：当邮寄 Log 文件时告诉 logrotate 需要执行的命令。
- -s,--state <statefile>：使用指定的状态文件。
- --usage：显示命令的基本用法。
- -v：开启 verbose 模式，显示命令执行的详细过程。

应用实例如下。

（1）管理指定的记录文件。

```
#logrotate /home/log.config
```

（2）强制执行文件维护操作。

```
# logrotate /etc/logrotate.conf
```

8.39 lsmod：显示 Linux 内核的模块信息

作用：lsmod 命令用来显示已载入系统的模块。

用法：lsmod

该命令无参数。

说明

执行 lsmod 命令，会列出所有已载入系统的模块。Linux 操作系统的核心具有模块化的特性，因此在编译核心时，无须把全部的功能都放入核心。可以将这些功能编译成一个个单独的模块，待需要时再分别载入。

应用实例如下。

显示内核加载的模块，使用如下命令：

```
# lsmod |more
Module                             Size    Used by
autofs4                            23749   2
hidp                               23105   2
l2cap                              29505   5 hidp
bluetooth                          53925   2 hidp,l2cap
video                              9269    0
sbs                                18533   0
pcnet32                            36805   0
i2c_core                           23745   2 i2c_ec,i2c_piix4
pcspkr                             7105    0
```

上面显示了当前系统中加载的模块，左边数第一列是模块名，第二列是该模块大小，第三列则是该模块使用的数量。如果后面为 unused，则表示该模块当前没在使用。如果后面为 autoclean，则该模块可以被 rmmod -a 命令自动清洗。rmmod -a 命令会将目前有 autoclean 的模块卸载，如果这时某个模块未被使用，则将该模块标记为 autoclean。

8.40 man：格式化和显示在线手册

作用：man 命令用来格式化和显示在线手册。

用法：man [-adfhktwW] [section] [-M path] [-P pager] [-S list] [-m system] [-p string] name...

主要选项如下。

- -a：man 默认在找到第一个符合的帮助文件后立即停止查找，该参数可强制显示所有符合的帮助文件。
- -B：指定 HTML 文件的浏览器。
- -c：即使是已经排版的文件，也强制执行重新排版。
- -C <config_file>：指定 man 的环境设置文件，默认文件是/etc/man.config。

- -d：仅显示排错的信息，而不显示手册页。
- -D：显示和打印出帮助信息及排错信息。
- -f：显示系统命令和工具程序的简单帮助，同 whatis 命令。
- -h：显示帮助信息。
- -H：指定将 HTML 文件作为文本文件的命令。
- -k：同命令 apropos。
- -K：在所有的帮助文件中查找包含关键字的帮助文件。
- -m system：根据给定的系统名字指定要搜索的手册页。
- -M <path>：指定查找手册页的搜索路径。
- -p <预处理程序>：指定要排版帮助文件前的处理程序，可选项有 eqn(e)、grap(g)、pic(p)、tbl(t)、vgrind(v)和 refer(r)。
- -P<pager>：指定浏览方式。该命令将重写 MANPAGER 的环境变量。在默认情况下，man 命令使用/user/bin/less -is。
- -S <section_list>：所有帮助主题都隶属于不同的小节，在 section_list 中小节之间以冒号“:”隔开，如 1:2:8:n。
  - Ø 小节 1：用户命令。
  - Ø 小节 2：系统调用。
  - Ø 小节 3：程序库调用。
  - Ø 小节 4：设备。
  - Ø 小节 5：文件格式。
  - Ø 小节 6：游戏。
  - Ø 小节 7：杂项。
  - Ø 小节 8：系统命令。
  - Ø 小节 9：Kernel 内部命令。
  - Ø 小节 n：Tcl 或 Tk 命令。
- -t：用/usr/bin/gruff -Tps -mandoc 来排版帮助文件，并输出到标准输出。
- -w,--path：只显示帮助主题的位置，不显示其内容，同 manpath 命令。
- -W：和参数-w 效果相同，分行显示文件名，该选项通常用于 Shell 命令中，如 man -aw man | xargs ls -l。

应用实例如下。

1）查找指定命令的帮助文件

例如，查找 ls 命令的手册页帮助，在命令行提示符下输入：

```
#man ls
```

2）显示系统命令和工具程序的简单帮助，而不显示数页的详细文档

例如，以该方式查询命令 which，在命令行提示符下输入：

```
#man -f which
```

3）只显示帮助主题的位置，不显示其内容

例如，显示 which 命令帮助文件所在的路径，在命令行提示符下输入：

```
#man -w which
```

8.41 manpath：设置 man 手册的查询路径

作用：manpath 命令用来设置 man 手册的查询路径。

用法：manpath 选项

主要选项如下。

- -c,--catpath：显示相对 cat 路径。
- -C,--config-file=文件：使用该用户设置文件。
- -d,--debug：输出调试信息。
- -g,--global：显示整个全局手册路径。

应用实例如下。

显示帮助文件的查找路径。

```
# manpath
/usr/kerberos/man:/usr/local/share/man:/usr/share/man/zh_CN.UTF-8:/usr/share/man/zh_CN:/usr/share/man
```

8.42 modinfo：显示内核信息

作用：modinfo 命令会显示 Kernel 模块的对象文件，以显示该模块的相关信息。

用法：modinfo [选项] [模块文件]

主要选项如下。

- -a 或--author：显示模块开发人员。
- -d 或--description：显示模块的说明。
- -h 或--help：显示 modinfo 的参数使用方法。
- -p 或--parameters：显示模块所支持的参数。
- -0 –NULL：使用'/0'字符分隔 field 值，而不是一个新行。对脚本比较有用。
- -F --field：仅在一行上显示 field 值，这对于脚本较为有用。常用的 field 有 author、description、licence、param、depends、alias、filename。
- -V 或--version：显示版本信息。

应用实例如下。

显示一个模块信息。

```
# modinfo sg
filename:    /lib/modules/2.6.18-8.el5xen/kernel/drivers/scsi/sg.ko
alias:       char-major-21-*
version:     3.5.34
license:     GPL
description: SCSI generic (sg) driver
author:      Douglas Gilbert
srcversion:  487FCED419CF55F7D5C9EDA
depends:     scsi_mod
vermagic:    2.6.18-8.el5xen SMP mod_unload 686 REGPARM 4KSTACKS gcc-4.1
parm:        scatter_elem_sz:scatter gather element size (default:
max(SG_SCATTER_SZ, PAGE_SIZE)) (int)
parm:        def_reserved_size:size of buffer reserved for each fd (int)
parm:        allow_dio:allow direct I/O (default: 0 (disallow)) (int)
```

8.43 modprobe：自动处理可载入模块

作用：modprobe（module probe）命令用于自动处理可载入模块。

用法：modprobe [选项] [--help] [模块文件] [符号名称=符号值]

主要选项如下。

- -v --verbose：显示程序在干什么，通常在出问题的情况下，modprobe 才显示信息。
- -C --config：重载默认配置文件（/etc/modprobe.conf 或/etc/modprobe.d）。
- -c --showconfig：输出配置文件并退出。
- -n --dry-run：可以和-v 选项一起使用，调试非常有用。
- -i --ignore-install --ignore-remove：该选项会使得 modprobe 忽略配置文件中的、在命令行上输入的 install 和 remove 命令。
- -q --quiet：一般情况下，modprobe 在删除或插入一个模块时，若没有找到则会提示错误。使用该选项，会忽略指定的模块，并不提示任何错误信息。
- -r --remove：该选项会导致 modprobe 去删除，而不是插入一个模块。通常没有理由去删除内核模块，除非是一些有 bug 的模块。用户的内核也不一定支持模块的卸载。
- -V --verssion：版本信息。
- -f --force：和同时使用 --force-vermagic、--force-modversion 一样。使用该选项是比较危险的。
- -l --list：列出所有模块。
- -a --all：插入所有命令行中的模块。
- -t --type：强制-l 显示 dirname 中的模块。
- -s --syslog：错误信息写入 syslog。

说明

modprobe 可载入指定的个别模块，或载入一组相依的模块。modprobe 会根据 depmod 所产生的相依关系，决定要载入哪些模块。若在载入过程中发生错误，modprobe 会卸载整组的模块。

应用实例如下。

加载软驱模块。

```
# modprobe -v floppy
```

8.44 pmap：显示程序的内存信息

作用：pmap 命令用来显示进程、程序的内存信息。

用法：pmap [选项] pids

主要选项如下。

- -x：显示扩展格式。
- -d：显示设备格式。
- -q：不显示 header/footer 行。
- -V：显示版本信息。
- -A：输出限制范围的结果。

应用实例如下。

（1）显示进程 PID 为 221 的详细信息。

```
# pmap -x 2835
2835:   -bash
Address   Kbytes     RSS    Anon  Locked Mode   Mapping
00285000     100       -       -       - r-x--  ld-2.5.so
0029e000       4       -       -       - r----  ld-2.5.so
0029f000       4       -       -       - rw---  ld-2.5.so
002a2000    1256       -       -       - r-x--  libc-2.5.so
003dc000       8       -       -       - r----  libc-2.5.so
003de000       4       -       -       - rw---  libc-2.5.so
003df000      12       -       -       - rw---    [anon ]
0040d000       8       -       -       - r-x--  libdl-2.5.so
0040f000       4       -       -       - r----  libdl-2.5.so
bfb3c000      84       -       -       - rw---    [stack ]
```

```
-------- ------- ------- ------- -------
total kB    1434       -       -       -
```

（2）查看进程占用的内存及使用地址空间，以显示设备的格式显示。

```
#pmap -d 19601
```

8.45 procinfo：显示系统状态

作用：显示系统状态。procinfo 命令从/proc 目录里读取相关数据，将数据妥善整理后输出到标准输出设备。

用法：procinfo [选项] [-F <输出文件>] [-n <间隔秒数>]

主要选项如下。

- -a：显示所有信息。
- -b：显示磁盘设备的区块数目，而非存取数目。
- -d：显示系统信息每秒间的变化差额，而非总和的数值。本参数必须配合“-f”参数使用。
- -D：此参数效果和指定“-d”参数类似，但内存和交换文件的信息为总和数值。
- -f：进入全画面的互动式操作界面。
- -F<输出文件>：把信息状态输出到文件保存起来，而非预设的标准输出设备。
- -h：在线帮助。
- -i：显示完整的 IRP 列表。
- -m：显示系统模块和外围设备等相关信息。
- -n<间隔秒数>：设置全画面互动模式的信息更新速度，单位以秒计算。
- -s：显示系统的内存、磁盘空间、IRP 和 DMA 等信息，此为预设值。
- -S：搭配参数“-d”或“-D”使用时，每秒都会更新信息，不论是否使用参数“-n”。
- -v：显示版本信息。

应用实例如下。

显示系统整体信息，使用如下命令：

```
# procinfo
Linux 2.6.18-8.el5xen (brewbuilder@ls20-bc2-14.build.redhat.com) (gcc
4.1.1 20070105 ) #1 SMP Fri Jan 26 14:42:21 EST 2007 1CPU [cjh.(none)]
Memory:    Total      Used      Free    Shared    Buffers
Mem:      299704    181172    118532         0     16972
Swap:     524280         0    524280

Bootup: Thu Aug 14 09:33:59 2008   Load average: 0.16 0.03 0.01 1/86 2988
user  :     0:00:44.71  0.7%  page in :           0
```

```
nice  :    0:00:00.00  0.0%  page out:      0
system:    0:01:13.07  1.1%  swap in :      0
idle  :    1:48:35.20  98.2%  swap out:     0
steal :    0:00:01.32  0.0%
uptime:    1:50:34.32     context :    249115

irq  1:     108 Phys-irq  i8042    irq 16:     0 Phys-irq  uhci_hcd:u
irq  6:      5 Phys-irq  floppy   irq 17:   5068 Phys-irq  eth0
irq  7:      0 Phys-irq  parport0   irq256:  294276 Dynamic-irq  timer0
irq  8:      6 Phys-irq  rtc     irq257:      0 Dynamic-irq  resched
irq  9:      0 Phys-irq  acpi     irq258:      0 Dynamic-irq  callfun
irq 12:     952 Phys-irq  i8042    irq259:     0 Dynamic-irq  xenbus
irq 14:   10790 Phys-irq  ide0     irq260:     0 Dynamic-irq  console
irq 15:   22909 Phys-irq  ide1
```

8.46 pwd：显示工作目录

作用：执行 pwd 命令可立刻得知自己目前所在的工作目录的绝对路径。

用法：pwd [选项]

主要选项如下。

- -L,--logical：显示当前目录。
- -P,--physical：显示当前目录的实际物理地址。

应用实例如下。

显示出完整的当前活动目录名称。

```
# pwd
/root
```

8.47 reboot：重新启动系统

作用：执行 reboot 命令可让系统停止运作，并重新开机。

用法：reboot [选项]

主要选项如下。

- -d：重新开机时不把数据写入记录文件/var/tmp/wtmp。本参数具有“-n”参数的效果。
- -n, --no-sync：在重启或关机时不做同步处理。
- -f,--force：不论主机处于何种状态均强制立即关机。

– -p,--poweroff：在关机时，关闭电源。
– -w,--wtmp-only：实际上并不重新启动或停止，只是写 wtmp 记录。
– -q,--quiet：只显示错误信息。

应用实例如下。

模拟重新启动系统。

```
#reboot –w
```

8.48 rlogin：远程登录

作用：rlogin 命令用于进行远程登录。执行 rlogin 命令开启终端机阶段操作，并登录远程主机。

用法：rlogin [选项] [-e <脱离字符>] [-l <用户名称>] [主机名称或 IP 地址]

主要选项如下。

– -8：允许输入 8 位字符数据。
– -e<脱离字符>：设置脱离字符。
– -E：滤除脱离字符。
– -l<用户名称>：指定要登录远程主机的用户名称。
– -L：使用 litout 模式进行远程登录阶段操作。

应用实例如下。

（1）远程登录主机。

```
# rlogin 192.168.40.135
```

（2）指定用户远程登录主机。

```
# rlogin 192.168.40.135 –l cjh
```

8.49 rmmod：删除模块

作用：执行 rmmod 命令，可删除不需要的模块。

用法：rmmod [选项]模块名称

主要选项如下。

– -a：删除所有目前不需要的模块。
– -s：把信息输出至 syslog 常驻服务，而非终端机界面。
– -f, --force：强制移除模块，使用此选项比较危险。
– -v：显示指令执行的详细信息。

说明：

执行 rmmod 命令，可删除不需要的模块。Linux 操作系统的核心具有模块化的特性，因此在编译核心时，无须把全部的功能都放入核心。可以将这些功能编译成一个个单独的模块，待有需要时再分别载入它们。

应用实例如下。

移除已挂载的模块 vfat，使用如下命令：

```
# rmmodvfat
```

8.50 rpm：软件包管理

作用：rpm 命令用来管理 RPM 软件包。

用法：rpm [选项]

主要选项如下。

- -a：查询所有套件。
- -b：<完成阶段><套件档>+或-t<完成阶段><套件档>+：设置包装套件的完成阶段，并指定套件档的文件名称。
- -c：只列出组态配置文件。本参数需配合“-l”参数使用。
- -d：只列出文本文件。本参数需配合“-l”参数使用。
- -e<套件档>：或--erase<套件档>：删除指定的套件。
- -f<文件>+：查询拥有指定文件的套件。
- -h 或--hash：套件安装时列出标记。
- -i：显示套件的相关信息。
- -i<套件档>或--install<套件档>：安装指定的套件档。
- -l：显示套件的文件列表。
- -p<套件档>+：查询指定的 RPM 套件档。
- -q：使用询问模式，当遇到任何问题时，rpm 命令会先询问用户。
- -R：显示套件的关联性信息。
- -s：显示文件状态。本参数需配合“-l”参数使用。
- -U<套件档>或--upgrade<套件档>：升级指定的套件档。
- -v：显示命令执行过程。
- -vv：详细显示命令执行过程，以便于排错。
- -addsign<套件档>+：在指定的套件里加上新的签名认证。
- --allfiles：安装所有文件。
- --allmatches：删除符合指定的套件所包含的文件。

- --badreloc：发生错误时，重新配置文件。
- --buildroot<根目录>：设置产生套件时，欲当作根目录的目录。
- --changelog：显示套件的更改记录。
- --checksig<套件档>+：检验该套件的签名认证。
- --clean：完成套件的包装后，删除包装过程中所建立的目录。
- --dbpath<数据库目录>：设置欲存放 RPM 数据库的目录。
- --dump：显示每个文件的验证信息。本参数需配合“-l”参数使用。
- --excludedocs：安装套件时，不要安装文件。
- --excludepath<排除目录>：忽略在指定目录里的所有文件。
- --force：强行置换套件或文件。
- --ftpproxy<主机名称或 IP 地址>：指定 FTP 代理服务器。
- --ftpport<通信端口>：设置 FTP 服务器或代理服务器使用的通信端口。
- --help：在线帮助。
- --httpproxy<主机名称或 IP 地址>：指定 HTTP 代理服务器。
- --httpport<通信端口>：设置 HTTP 服务器或代理服务器使用的通信端口。
- --ignorearch：不验证软件包架构。
- --ignoreos：不验证软件包操作系统。
- --ignoresize：安装前不检查磁盘空间是否足够。
- --includedocs：安装套件时，一并安装文件。
- --initdb：确认有正确的数据库可以使用。
- --justdb：更新数据库，但不变动任何文件。
- --nobulid：不执行任何完成阶段。
- --nodeps：不验证套件档的相互关联性。
- --nofiles：不验证文件的属性。
- --nogpg：略过所有 GPG 的签名认证。
- --nomd5：不使用 MD5 编码演算确认文件的大小与正确性。
- --nopgp：略过所有 PGP 的签名认证。
- --noorder：不重新编排套件的安装顺序，以便满足其彼此间的关联性。
- --noscripts：不执行任何安装 Script 文件。
- --notriggers：不执行该套件包装内的任何 Script 文件。
- --oldpackage：升级旧版本的套件。
- --percent：安装套件时显示完成百分比。
- --pipe<执行命令>：建立管道，把输出结果转换为该执行命令的输入数据。
- --prefix<目的目录>：若重新配置文件，就把文件放到指定的目录下。
- --provides：查询该套件所提供的兼容度。
- --queryformat<档头格式>：设置档头的表示方式。
- --querytags：列出可用于档头格式的标签。
- --rcfile<配置文件>：使用指定的配置文件。

- --rebulid<套件档>：安装原始代码套件，重新产生二进制文件的套件。
- --rebuliddb：以现有的数据库为主，重建一份数据库。
- --recompile<套件档>：此参数的效果和指定“--rebulid”参数类似，但不产生套件档。
- --relocate<原目录>=<新目录>：把本来会放到原目录下的文件改放到新目录下。
- --replacefiles：强行置换文件。
- --replacepkgs：强行置换套件。
- --requires：查询该套件所需要的兼容度。
- --resing<套件档>+：删除现有认证，重新产生签名认证。
- --rmsource：完成套件的包装后，删除原始代码。
- --rmsource<文件>：删除原始代码和指定的文件。
- --root<根目录>：设置要当作根目录的目录。
- --scripts：列出安装套件的 Script 的变量。
- --setperms：设置文件的权限。
- --setugids：设置文件的拥有者和所属群组。
- --short-circuit：直接略过指定完成阶段的步骤。
- --sign：产生 PGP 或 GPG 的签名认证。
- --target=<安装平台>+：设置产生的套件的安装平台。
- --test：仅做测试，并不真正地安装套件。
- --timecheck<检查秒数>：设置检查时间的计时秒数。
- --triggeredby<套件档>：查询该套件的包装者。
- --triggers：展示套件档内的包装 Script。
- --verify：此参数的效果和指定“-q”参数相同。
- --version：显示版本信息。
- --whatprovides<功能特性>：查询该套件对指定的功能特性所提供的兼容度。
- --whatrequires<功能特性>：查询该套件对指定的功能特性所需要的兼容度。

说明

rmp 原本是 Red Hat Linux 发行版专门用来管理 Linux 各项套件的程序，由于它遵循 GPL 规则且功能强大，因而广受欢迎，逐渐受到其他发行版的采用。RPM 套件管理方式的出现，让 Linux 易于安装、升级，间接地提升了 Linux 的适用度。

应用实例如下。

1）安装软件包

安装 foo-1.0-1.i386.rpm 软件包。

```
#rpm -ivh foo-1.0-1.i386.rpm
```

如果安装成功，系统会显示软件包的名称，然后在软件包安装时在屏幕上打印“#”显示安装的进度，显示如下信息：

```
Preparing... ########################################### [100%]
1:foo########################################### [100%]
```

如果某软件包的同一版本已经安装，系统会显示如下信息：

```
Preparing...########################################### [100%]
package foo-1.0-1 is already installed
```

如果在软件包已安装的情况下仍打算安装同一版本的软件包，可以使用“--replacepkgs”选项忽略错误。在 foo-1.0-1.i386.rpm 已安装的情况下仍旧安装该软件包。

```
#rpm -ivh --replacepkgs foo-1.0-1.i386.rpm
```

2）软件包冲突

如果试图安装的软件包中包含已被另一个软件包或同一软件包的早期版本安装了的文件，系统会显示如下信息：

```
Preparing... ########################################### [100%]
file /usr/bin/foo from install of foo-1.0-1 conflicts with file from package
bar-2.0.20
```

使用“--replacefiles”选项可以忽略这个错误。

```
#rpm -ivh --replacefiles foo-1.0-1.i386.rpm
```

3）刷新软件包

刷新 foo-1.2-1.i386.rpm 软件包。

```
#rpm -Fvh foo-1.2-1.i386.rpm
```

使用 RPM 刷新软件包时，系统会比较指定的软件包的版本和系统上已安装的版本。但 RPM 的刷新选项处理的版本比已安装的版本更新，它就会升级到更新的版本。然而，如果某软件包先前没有安装，RPM 的刷新选项将不会安装该软件包。这和 RPM 的升级选项不同，因为不管该软件包的早期版本是否已被安装，升级选项都会安装该软件包。

4）查询软件包

使用“rpm -q”命令查询软件包安装的信息，如查询系统内所有已安装的 RPM 软件包。

```
 # rpm -qa|more
glib2-2.12.3-2.fc6
libart_lgpl-2.3.17-4
libattr-2.4.32-1.1
cyrus-sasl-lib-2.1.22-4
beecrypt-4.1.2-10.1.1
libIDL-0.8.7-1.fc6
dvd+rw-tools-7.0-0.el5.3
flac-1.1.2-27
```

```
eject-2.1.5-4.2.el5
patch-2.5.4-29.2.2
```

5）校验软件包

校验软件包将检查从软件包安装的文件和原始软件包中的同一文件的信息，它校验每个文件的大小、MD5 值、权限、类型、所有者和组群。如校验所有在 vsftpd 软件包内的文件。

```
# rpm -V vsftpd
```

校验所有安装的软件包。

```
#rpm -Va
S.5....T c /etc/xml/catalog
S.5....T c /usr/share/sgml/docbook/xmlcatalog
S.5....T c /etc/sysconfig/system-config-securitylevel
.......T c /etc/kdump.conf
.....UG. c /var/arpwatch/arp.dat
.M....G. /tmp
…
```

根据 dvdrp-0.98-1.el5.rf.i386.rpm 软件包进行校验。

```
# rpm -Vp dvdrp-0.98-1.el5.rf.i386.rpm
warning: dvdrp-0.98-1.el5.rf.i386.rpm: Header V3 DSA signature: NOKEY,
keyID 9e6
```

说明

如果一切都被校验正确，屏幕上就不会显示输出。如果出现矛盾，它们就会被显示。输出的格式为包含 4 个字符的字符串（c 代表配置文件）和文件名称。这 4 个字符中的每个字符都代表一种文件属性的比较结果，所比较的是文件的属性和 RPM 数据库中记录的属性。单用一个“.”意味着测试通过。下列字符代表某类测试失败。

- 5：MD5 校验和。
- S：文件大小。
- L：符号链接。
- T：文件修改时间。
- D：设备。
- U：用户。
- G：组群。
- M：模式（包括权限和文件类型）。
- ?：不可读文件。

8.51 shutdown 命令：系统关机命令

作用：shutdown 命令可以关闭所有程序，并依用户的需要，进行重新开机或关机的动作。

用法：shutdown [选项] [-t 秒数] [时间] [警告信息]

主要选项如下。

- -r：关闭并重新启动此计算机。
- -h：完全终止操作系统，等价于 poweroff 命令。
- -H：完全终止操作系统，关闭系统。
- -P：完全终止操作系统，重启系统。
- -c：取消一个已经运行的 shutdown。
- -k：发送警告信息，而不关机。
- --no-wall：不发送警告信息。
- -t<秒数>：送出警告信息和删除信息之间要延迟多少秒。
- [时间]：设置多长时间后执行 shutdown 命令。
- [警告信息]：要传送给所有登录用户的信息。

应用实例如下。

（1）晚上 11：00 关机。

```
#shutdown –h 23:00
```

（2）系统马上关机并且不重新启动。

```
shutdown -h now
```

（3）立即重启。

```
shutdown -r now
```

（4）希望在 20:13 关闭计算机。

```
shutdown -h 20:13
```

（5）指定 5 分钟后关机，同时送出警告信息给登录用户。

```
shutdown +5 "System will shutdown after 5 minutes"
```

8.52 suspend：暂停执行 Shell

作用：挂起 Shell 的执行，直到收到一个 SIGCONT 信号。

用法：suspend [-f]

选项-f 表示如果这是一个登录 Shell，那么不要提示，直接挂起。返回值是 0，除非 Shell 是登录 Shell 并且没有指定-f，或者没有启用作业控制。

应用实例如下。

强制挂起当前 Shell，使用如下命令：

```
# suspend -f
```

8.53 nproc：打印当前进程可用的处理器数

作用：打印当前进程可用的处理器数，此数目可能小于实际工作数。

用法：nproc [选项]...

主要选项如下。

- --all：打印所拥有的处理器数目。
- --ignore=N：如果有可能，忽略 *N* 个处理器。

应用实例如下。

打印当前进程可用的处理器数。

```
#nproc
1
```

8.54 tload：显示系统负载

作用：显示系统负载。tload 命令使用 ASCII 字符简单地以文字模式显示系统负载状态。假设不给予终端机编号，则会在执行 tload 命令的终端机上显示负载情形。

用法：tload [-V] [-d <间隔秒数>] [-s <刻度大小>] [终端机编号]

主要选项如下。

- -d<间隔秒数>：设置 tload 检测系统负载的间隔时间，单位以秒计算。
- -s<刻度大小>：设置图表的垂直刻度大小，单位以列计算。

应用实例如下。

显示系统负载。

```
# tload
1、22,1、12,0.65
```

8.55 uname：显示系统信息

作用：显示系统信息。uname 可显示计算机及操作系统的相关信息。

作用：uname [选项] [--help] [--version]

主要选项如下。

- a, --all：以如下次序输出所有信息。若-p 和-i 的探测结果不可知则被省略。
- -s, --kernel-name：输出内核名称。
- -n, --nodename：输出网络节点上的主机名。
- -r, --kernel-release：输出内核发行号。
- -v, --kernel-version：输出内核版本。
- -m, --machine：输出主机的硬件架构名称。
- -p, --processor：输出处理器类型或“unknown”。
- -i, --hardware-platform：输出硬件平台或“unknown”。
- -o, --operating-system：输出操作系统名称。

应用实例如下。

显示主机全部的信息，使用如下命令：

```
# uname -a
Linux cjh 2.6.18-8.el5xen #1 SMP Fri Jan 26 14:42:21 EST 2007 i686 i686 i386 GNU/Linux
```

8.56 authconfig：配置系统的认证信息

作用：authconfig 命令用来配置系统的认证信息。

用法：authconfig [选项] <--update|--test|--probe>

主要选项如下。

- --enableshadow, --useshadow：默认启用屏蔽密码。
- --disableshadow：默认禁用屏蔽密码。
- --enablemd5, --usemd5：默认启用 MD5 密码。
- --disablemd5：默认禁用 MD5 密码。
- --enablenis：默认使用 NIS 用户信息。
- --disablenis：默认禁用 NIS 用户信息。
- --nisdomain=<domain>：默认的 NIS 域。
- --nisserver=<server>：默认的 NIS 服务器。
- --enableldap：默认使用 LDAP 用户信息。
- --disableldap：默认禁用 LDAP 用户信息。
- --enableldapauth：默认使用 LDAP 验证。
- --disableldapauth：默认禁用 LDAP 验证。
- --ldapserver=<server>：默认的 LDAP 服务器。
- --ldapbasedn=<dn>：默认的基于 DN 的 LDAP。

- --enableldaptls, --enableldapssl：LDAP 使用 TLS。
- --disableldaptls, --disableldapssl：LDAP 不使用 TLS。
- --ldaploadcacert=<URL>：从这个 URL 加载 CA 证书。
- --enablesmartcard：默认启用智能卡验证。
- --disablesmartcard：默认禁用智能卡验证。
- --enablerequiresmartcard：默认需要使用验证智能卡验证。
- --disablerequiresmartcard：默认不需要使用智能卡验证。
- --smartcardmodule=<module>：默认使用的智能卡模块。
- --smartcardaction=<0=锁定|1=忽略>：删除智能卡时进行的操作。
- --enablekrb5：默认启用 Kerberos 验证。
- --disablekrb5：默认禁用 Kerberos 验证。
- --krb5kdc=<server>：默认的 Kerberos KDC。
- --krb5adminserver=<server>：默认的 Kerberos 管理服务器。
- --krb5realm=<realm>：默认的 Kerberos 域。
- --enablekrb5kdcdns：启用 DNS 来查找 Kerberos KDC。
- --disablekrb5kdcdns：禁用 DNS 来查找 Kerberos KDC。
- --enablekrb5realmdns：启用 DNS 来查找 Kerberos 域。
- --disablekrb5realmdns：禁用 DNS 来查找 Kerberos 域。
- --enablesmbauth：默认启用 SMB 验证。
- --disablesmbauth：默认禁用 SMB 验证。
- --smbservers=<server>：用来验证的服务器名称。
- --smbworkgroup=<workgroup>：验证服务器所在的组。
- --enablewinbind：默认启用 winbind 来获取用户信息。
- --disablewinbind：默认禁用 winbind 来获取用户信息。
- --enablewinbindauth：默认启用 winbind 进行验证。
- --disablewinbindauth：默认禁用 winbind 进行验证。
- --smbsecurity=<user|server|domain|ads>：samba 和 winbind 使用的安全模式。
- --smbrealm=<realm>：当 security=ads 时，samba 和 winbind 的默认域。
- --smbidmapuid=<lowest-highest>：winbind 分配域或 ads 用户的 UID 范围。
- --smbidmapgid=<lowest-highest>：winbind 分配域或 ads 用户的 GID 范围。
- --winbindseparator=<\>：如果没有启用 winbindusedefaultdomain，用来分隔 winbind 创建的用户名中域和用户部分的字符。
- --winbindtemplatehomedir=</home/%D/%U>：winbind 创建的用户将会作为主目录的目录。
- --winbindtemplateprimarygroup=<nobody>：winbind 创建的用户将会作为主要组群的组群。
- --winbindtemplateshell=</bin/false>：winbind 创建的用户将会作为 Shell 的 Shell。
- --enablewinbindusedefaultdomain：配置 winbind 以假定在用户名中没有域的用户是域用户。
- --disablewinbindusedefaultdomain：配置 winbind 以假定在用户名中没有域的用户不是域用户。

- --winbindjoin=<Administrator>：立即作为系统管理员加入 winbind 域或 ads 域。
- --enablewins：启用 wins 进行主机名解析。
- --disablewins：禁用 wins 进行主机名解析。
- --enablehesiod：默认启用 hesiod 来获取用户信息。
- --disablehesiod：默认禁用 hesiod 来获取用户信息。
- --hesiodlhs=<lhs>：默认的 hesiod LHS。
- --hesiodrhs=<rhs>：默认的 hesiod RHS。
- --enablecache：默认启用缓存用户信息。
- --disablecache：默认禁用缓存用户信息。
- --enablelocauthorize：本地用户可以使用本地授权。
- --disablelocauthorize：通过远端服务授权本地用户。
- --enablesysnetauth：通过网络服务认证的系统账号。
- --disablesysnetauth：仅通过本地文件认证的系统账号。
- --nostart：不要启动或停止 portmap、ypbind 和 nscd。
- --test：不更新配置文件，只输出新的设置。
- --update, --kickstart：与--test 相反，以修改过的设置升级配置文件。
- --updateall：更新所有配置文件。
- --probe：探测网络默认值并输出它们。

应用实例如下。

更新所有配置文件。

```
#authconfig –updateall
```

8.57 declare：显示或者设定 Shell 变量

作用：declare 命令用于声明 Shell 变量。

用法：declare [+/-] [rxi] [变量名称＝设置值]

　　　declare -f

主要选项如下。

- +/-："-"用来指定变量的属性，"+"则用来取消为变量所设的属性。
- -f：仅显示函数。
- -r：将变量设置为只读。
- -x：指定的变量会成为环境变量，可供 Shell 以外的程序使用。
- -i："设置值"可以是数值、字符串或运算式。
- -a：声明数组变量。

应用实例如下。

显示当前 Shell 变量，使用如下命令：

```
# declare
BASH=/bin/bash
BASH_ARGC=()
BASH_ARGV=()
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="3" [1]="1" [2]="17" [3]="1" [4]="release"
[5]="i686-redhat-linux- gnu")
BASH_VERSION='3.1.17(1)-release'
CLASSPATH=.:/usr/java/jdk1.5.04/lib/dt.jar:/usr/java/jdk1.5.04/lib/tools.jar
COLORS=/etc/DIR_COLORS.xterm
COLUMNS=144
CVS_RSH=ssh
…
```

8.58 export：设置或者显示环境变量

作用：export 命令用来设置或显示环境变量。

用法：export [-fnp] [变量名称]=[变量设置值]

主要选项如下。

- -f：代表变量名称中的函数名称。
- -n：删除指定的变量。变量实际上并未删除，只是不会输出到后续命令的执行环境中。
- -p：列出 Shell 赋予程序的所有环境变量。

> **说明**
>
> 在 Shell 中执行程序时，Shell 会提供一组环境变量。export 可新增、修改或删除环境变量，供后续执行的程序使用。export 的效力仅次于该次登录操作。

应用实例如下。

显示当前所有 Shell 变量。

```
# export -p
Declare-x
CLASSPATH=".:/usr/java/jdk1.5.04/lib/dt.jar:/usr/java/jdk1.5.04/lib/tools.jar"
declare -x CVS_RSH="ssh"
declare -x GRAILS_HOME="/grails"
declare -x G_BROKEN_FILENAMES="1"
declare -x HISTSIZE="1000"
```

8.59 hostid：打印出当前主机的标识

作用：hostid 命令用于显示或者打印出当前主机的标识（十六进制）。

用法：hostid 选项

主要选项如下。

- --help：显示此帮助信息并退出。
- --version：输出版本信息并退出。

应用实例如下。

显示当前主机的标识。

```
# hostid
a8c08728
```

8.60 insmod：载入模块

作用：insmod 命令用来加载模块。

用法：insmod [-选项] [-o <模块名称>] [模块文件] [符号名称=符号值]

主要选项如下。

- -f：不检查目前 Kernel 版本与模块编译时的 Kernel 版本是否一致，强制将模块载入。
- -k：将模块设置为自动卸除。
- -m：输出模块的载入信息。
- -o<模块名称>：指定模块的名称，可使用模块文件的文件名。
- -p：测试模块是否能正确地载入 Kernel。
- -s：将所有信息记录在系统记录文件中。
- -v：执行时显示详细的信息。
- -x：不要汇出模块的外部符号。
- -X：汇出模块所有的外部符号，此为预设置。

> **说明**
>
> Linux 有许多功能是通过模块的方式在需要时才载入 Kernel 的。如此可使 Kernel 较为精简，进而提高效率，以及保有较大的弹性。这类可载入的模块通常是设备驱动程序。

应用实例如下。

加载一个模块。

```
# insmod hdc.o
```

8.61 rdate：显示其他主机的日期与时间

作用：执行 rdate 命令，向其他主机询问系统时间并显示出来。

用法：rdate [-s] [-p] [-u] [-l] [-t sec] <host> ..

主要选项如下。

– -u：传输协议使用 UDP 协议。
– -l：使用 syslog 显示错误信息。
– -t<时间>：设置超时时间。
– -p：显示远端主机的日期与时间。
– -s：把从远端主机收到的日期和时间回存到本地主机的系统时间。

应用实例如下。

显示远端主机的日期与时间。

```
# rdate -p 192.168.40.135
```

8.62 runlevel：显示执行等级

作用：runlevel 命令用来显示当前执行等级。

用法：runlevel

系统的级别如下。

– 0：停机。
– 1：单用户模式。
– 2：多用户模式，无网络。
– 3：完全的多用户模式。
– 4：用户自定义。
– 5：图形界面多用户模式。
– 6：重启。

应用实例如下。

显示当前执行等级。

```
# runlevel
N 5
```

8.63 set：设置 Shell

作用：set 命令用来设置 Shell。set 命令能设置所使用 Shell 的执行方式，可依照不同的需求来进行设置。

用法：set [+-abCdefhHklmnpPtuvx]

主要选项如下。

- -a：标识已修改的变量，以供输出至环境变量。
- -b：使被中止的后台程序立刻回到执行状态。
- -C：转向所产生的文件无法覆盖已存在的文件。
- -d：Shell 预设会用杂凑表记忆使用过的命令，以加速命令的执行，使用-d 参数可取消。
- -e：若命令传回值不等于 0，则立即退出 Shell。
- -f：取消使用通配符。
- -h：自动记录函数的所在位置。
- -H Shell：可利用“!”加<命令编号>的方式来执行 history 中记录的命令。
- -k：命令所给的参数都会被视为此命令的环境变量。
- -l：记录 for 循环的变量名称。
- -m：使用监视模式。
- -n：只读取命令，而不实际执行。
- -p：启动优先顺序模式。
- -P：启动-P 参数后，执行命令时，会以实际的文件或目录来取代符号链接。
- -t：执行完随后的命令，即退出 Shell。
- -u：当执行时用到未定义过的变量，则显示错误信息。
- -v：显示 Shell 所读取的输入值。
- -x：执行命令后，会先显示该命令及其下的参数。
- +<参数>：取消某个 set 曾启动的参数。

应用实例如下。

（1）输出当前 Shell 变量。

```
#set
BASH=/bin/bash
BASH_ARGC=()
BASH_ARGV=()
BASH_LINENO=()
BASH_SOURCE=()
…
```

（2）设置名为 TEST^1 的环境变量。

```
#set testVar=test^^1
```

（3）设置名为 TEST&1 的环境变量。

```
#set testVar=test^&1
```

（4）设置 C Shell 路径。

C Shell 的路径是在用户的＄HOME/.cshrc 文件中设置的。

```
set path=(./usr/bin ＄ home/bin)
```

8.64 setenv：查询或显示环境变量

作用：setenv（set environment variable）是 TSCH 中用于查询或显示环境变量的命令。

用法：setenv [变量名称] [变量值]

应用实例如下。

设置一个环境变量。

```
setenv LDFLAGS "-L/usr/local/lib"
```

> **说明**
> setenv 是 TCSH 中的命令，BASH 是不用的，BASH 的相应命令是 export。

8.65 setserial：设置或显示串口的相关信息

作用：setserial 可用来设置或显示串口目前的设置。

作用：setserial [选项] [设备] [串口参数]

主要选项如下。

- -a：显示详细信息。
- -b：显示摘要信息。
- -g：显示串口的相关信息。
- -G：以命令列表的格式显示信息。
- -q：执行时显示较少的信息。
- -v：执行时显示较多的信息。
- -V：显示版本信息。
- -z：设置前先将所有的标记归零。

应用实例如下。

显示串口的详细信息。

```
# setserial -g –a
```

8.66 setup：设置公用程序

作用：设置公用程序。

用法：setup

说明

setup 是一个设置公用程序、提供图形界面的操作方式。在 setup 中可设置如下 7 类选项。

- 登录认证方式。
- 键盘组态设置。
- 鼠标组态设置。
- 开机时所要启动的系统服务。
- 声卡组态设置。
- 时区设置。
- X Window 组态设置。

运行 setup 命令进入交互界面，如图 8-4 所示。

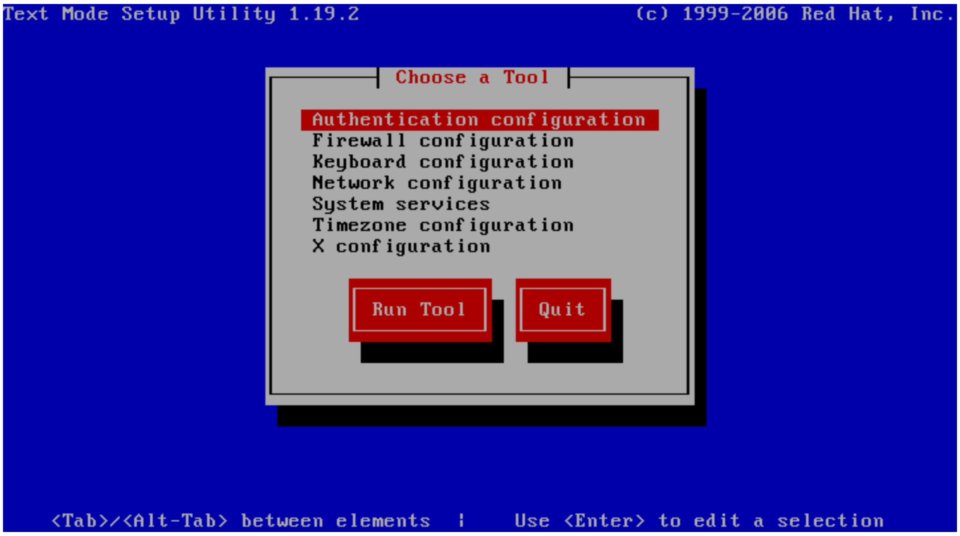


图 8-4　setup 界面

说明

不同版本 Linux 的 setup 命令有所差异。

8.67 symlinks：维护符号链接的工具程序

作用：symlinks 命令用来维护符号链接的工具程序。

用法：symlinks [-cdrstv] dirlist

主要选项如下。

- -c：将绝对路径的符号链接转换为相对路径（在相同文件系统内）。
- -d：移除 dangling 类型的符号链接。
- -r：检查目录下的子目录内的所有符号链接（在相同文件系统内）。
- -s：检查 lengthy 类型的符号链接。
- -t：该参数和-c 参数一起使用，显示如何将绝对路径的符号链接转换为相对路径，但是不会进行实际的转换。
- -v：显示所有类型的符号链接，默认不会显示 relative 类型的符号链接。

说明

symlinks 可检查目录中的符号链接，并显示符号链接类型。以下为 symlinks 可判断的符号链接类型。

- absolute：符号链接使用了绝对路径。
- dangling：原始文件已经不存在。
- lengthy：符号链接的路径中包含了多余的“../”。
- messy：符号链接的路径中包含了多余的“/”。
- other_fs：原始文件位于其他文件系统中。
- relative：符号链接使用了相对路径。

应用实例如下。

（1）检查并列出指定目录下所有符号链接。

例如，以默认方式检查 tmp/目录下所有的符号链接并显示其符号链接的类型。

```
# symlinks /tmp
```

（2）检查目录下的子目录内的所有符号链接，并列出符号链接的类型。

例如，列出 tmp/目录及其子目录下的所有符号链接，可以通过选项-r 实现。

```
# symlinks -r /tmp
```

8.68 swatch：系统监控程序

作用：swatch 命令是用来进行系统监控的程序。

用法：swatch [-c 设定文件] [-f 记录文件] [-r 时间] [-t 记录文件]

主要选项如下。

- -c<设定文件>：用非默认模式指定设定文件。
- -f<记录文件>：检查指定的记录文件，完成检查后终止对该文件的监控。
- -r<时间>：设定重新启动的时间。
- -t<记录文件>：检查指定的记录文件，并且监控加入记录文件中的后续记录。

8.69 sync：将内存缓冲区内的数据写入磁盘

作用：在 Linux 系统中，当数据需要存入磁盘时，通常会先放到缓冲区内，等到适当的时刻再写入磁盘，如此可提高系统的执行效率。强迫将已更改的数据写入磁盘，并更新超级块。

用法：sync [选项]

主要选项如下。

- --help：显示帮助。
- --version：显示版本信息。

应用实例如下。

将内存缓冲区内的数据写入磁盘。

```
#sync
```

8.70 startx：启动 X Window

作用：startx（start X Window）为启动 X Window 的脚本文件，实际上启动 X Window 的程序为 xinit。

用法：startx

应用实例如下。

启动 X Window，使用如下命令：

```
#startx
```

或者：

```
#xinit
```

8.71 sysctl：设置系统核心参数

作用：设置和显示在/proc/sys 目录中的内核参数。可以使用 sysctl 来设置或重新设置联网功能，如 IP 转发、IP 碎片去除及源路由检查等。

用法：sysctl [-n] [-e] -w variable=value

sysctl [-n] [-e] -p

sysctl [-n] [-e] -a

主要选项如下。

- -w：临时改动某个指定参数的值。
- -a：显示所有的系统参数。
- -p：从指定的文件加载系统参数，如不指定则从/etc/sysctl.conf 中加载。
- -n：禁止打印参数名称。
- -e：对于不知道的参数，忽略其错误。
- -N：只打印名称。
- -q：不打印参数值到标准输出。
- -a：显示所有的系统参数。
- -A：以表格形式显示所有值。

应用实例如下。

启动 IP 转发，使用如下命令：

```
#sysctl -w net.ipv4.ip_forward=1
```

> **说明**
> 如果系统重启，所设置的值就会丢失。

8.72 timeconfig：设置时区

作用：这是 Red Hat 公司遵循 GPL 规则所开发的程序，它具有互动式操作界面，可以轻易地利用方向键和空格键等设置系统时间所属的时区，是 RHEL 老版本的工具。

用法：timeconfig [--arc] [--back] [--test] [--utc] [时区名称]

主要选项如下。

- --arc：使用 Alpha 硬件结构的格式存储系统时间。
- --back：在互动式界面里，显示 Back 按钮而非 Cancel 按钮。
- --test：仅做测试，并不真正地改变系统的时区。
- --utc：把硬件时钟上的时间视为 CUT，有时也称为 UTC 或 UCT。

应用实例如下。

运行没有参数的 timeconfig 命令进入交互界面，如图 8-5 所示。

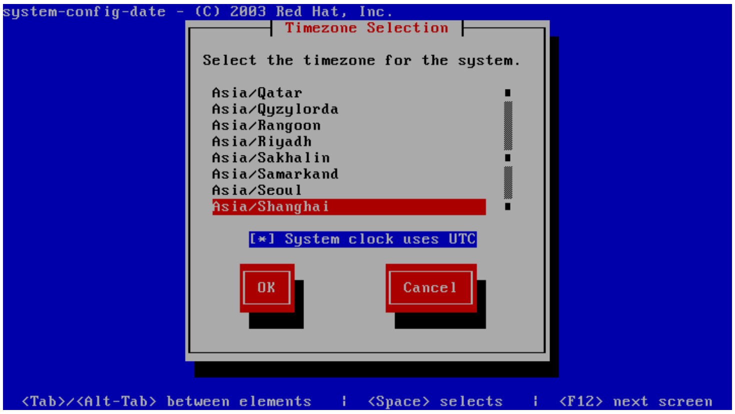


图 8-5　timeconfig 界面

8.73 ulimit：控制 Shell 程序的资源

作用：ulimit 为 Shell 内建命令，可用来控制 Shell 执行程序的资源。

用法：ulimit [-SHacdefilmnpqrstuvx] [限制]

主要选项如下。

- -a：显示目前资源限制的设定。
- -c<core 文件上限>：设定 core 文件的最大值，单位为区块。
- -d<数据节区大小>：程序数据节区的最大值，单位为 KB。
- -f<文件大小>：Shell 所能建立的最大文件，单位为区块。
- -H：设定资源的硬性限制，也就是管理员所设下的限制。
- -m<内存大小>：指定可使用内存的上限，单位为 KB。
- -n<文件数目>：指定同一时间最多可开启的文件数。
- -p<缓冲区大小>：指定管道缓冲区的大小，单位为 512 字节。
- -s<堆叠大小>：指定堆叠的上限，单位为 KB。
- -S：设定资源的弹性限制。
- -t<CPU 时间>：指定 CPU 使用时间的上限，单位为秒。
- -u<程序数目>：指定用户最多可开启的程序数目。
- -v<虚拟内存大小>：指定可使用的虚拟内存上限，单位为 KB。

应用实例如下。

（1）显示当前资源限制。

```
# ulimit -a
core file size                      (blocks, -c)        0
```

```
data seg size              (kbytes, -d)      unlimited
max nice                           (-e)      0
file size                  (blocks, -f)      unlimited
pending signals                    (-i)      4808
max locked memory          (kbytes, -l)      32
max memory size            (kbytes, -m)      unlimited
open files                         (-n)      1024
pipe size               (512 bytes, -p)      8
POSIX message queues        (bytes, -q)      819200
max rt priority                    (-r)      0
stack size                 (kbytes, -s)      10240
cpu time                  (seconds, -t)      unlimited
max user processes                 (-u)      4808
virtual memory             (kbytes, -v)      unlimited
file locks                         (-x)      unlimited
```

输出说明如下。

输出的每一行由资源名字、单位、ulimit 命令的参数、软限制组成。

详细解释如下。

- core file size：core 文件的最大值为 100 Blocks。
- data seg size：进程的数据段可以任意大。
- file size：文件可以任意大。
- pending signals：最多有 2047 个待处理的信号。
- max locked memory：一个任务锁住的物理内存的最大值为 32KB。
- max memory size：一个任务常驻物理内存的最大值。
- open files：一个任务最多可以同时打开 1024 个文件。
- pipe size：管道的最大空间为 4096 字节。
- POSIX message queues：POSIX 的消息队列的最大值为 819 200 字节。
- stack size：进程的栈的最大值为 8192 字节。
- cpu time：进程使用的 CPU 时间。
- max user processes：当前用户同时打开的进程（包括线程）的最大个数为 2047。
- virtual memory：没有限制进程的最大地址空间。
- file locks：所能锁住的文件的最大个数没有限制。

（2）将文件大小极限设置为 100KB。

```
#ulimit -f 100
```

8.74 unalias：删除别名

作用：unalias 为 Shell 内建命令，可删除别名设置。

用法：unalias [-a] [别名]

主要选项如下。

-a：删除全部的别名。

应用实例如下。

删除全部的别名。

```
# unalias –a
```

8.75 unset：删除变量或函数

作用：unset 为 Shell 内建命令，可删除变量或函数。

用法：unset [-fv] [变量或函数名称]

主要选项如下。

– -f：仅删除函数。
– -v：仅删除变量。

应用实例如下。

删除一个环境变量 PWD。

```
#unset PWD
```

8.76 up2date：软件包升级

作用：up2date 是 RHEL 老版本的命令行的软件包升级工具。

用法：up2date [选项]

主要选项如下。

– --configure：配置更新代理的选项。
– -d, --download：只下载软件包，即便配置文件说要安装。
– --src：下载源软件包及二进制 RPM。
– --nosrc：不要下载源软件包（如果设置了，则取代配置设置）。
– --dbpath=<dir>：指定一个可以发现另一个 RPM 数据库的路径。
– -f, --force：强制安装软件包，取代软件包和文件跳过列表。
– -i, --install：安装软件包，即便配置文件说要只下载。

- --justdb：只将软件包添加到数据库中，不把它们安装到文件系统中。
- -l, --list：列举可用的软件包以便检索。
- -h, --help：显示帮助屏幕。
- -?, --usage：简略描述选项。
- -k, --packagedir=STRING：要在其中查询软件包的用冒号分隔的目录路径。
- --nosig：不要使用 GPG 来检查软件包签名。
- --nox：不要试图使用 X。
- -p, --packages：更新与 Red Hat 网络上的这个系统档案相关联的软件包。
- --showall：列举所有可下载的软件包。
- --solvedeps=<deps>;：搜寻、下载并安装所需的软件包来解决这些依赖关系。
- --tmpdir=<directory>;：存储临时文件或 RPM 的地方。
- -u, --update：用所有相关的软件包更新系统。
- --version：显示程序版本信息。
- -v, --verbose：显示附加的输出。
- --show-channels：与软件包名称一起在恰当的地方显示频道名称。
- --whatprovides=<deps>;：显示用逗号分隔的有依赖关系的软件包。
- --channel=STRING：指定哪些频道要被频道标签使用。
- --gpg-flags：显示 GPG 会被引发的标志（钥匙圈等）。
- --nodownload：不下载软件包，只用来测试。

应用实例如下。

显示所有可以使用的更新包。

```
#up2date –l
```

8.77 uptime：告知系统运行了多长时间

作用：uptime 命令给出一行显示的信息，包括当前时间、系统运行了多长时间、当前登录的用户有多少，以及前 1 分钟、5 分钟和 15 分钟系统的平均负载。

用法：uptime 选项

主要选项如下。

- -p, --pretty：使用 pretty 显示格式，不显示负载。
- -s, --since：系统启动后运行的时间。

应用实例如下。

显示系统运行时间和负载，使用如下命令：

```
# uptime
 14:20:57 up4:46,4 users,load average: 0.01, 0.02, 0.01
```

8.78 mouseconfig：设置鼠标相关参数

作用：mouseconfig 为鼠标设置程序，可自动设置相关参数，用户也可以利用所提供的互动模式自行设置鼠标，是 RHEL 老版本的工具。

用法：mouseconfig [选项] [--device <连接端口>] [鼠标类型]

主要选项如下。

- --back：在设置画面上显示 Back 按钮，取代预设的 Cancel 按钮。
- --device<连接端口>：指定硬件连接端口。可用的选项有 ttyS0、ttyS1、ttyS2、ttyS3 与 orpsaux。
- --emulthree：将二钮鼠标模拟成三钮鼠标。
- --help：显示帮助及所有支持的鼠标类型。
- --expert：程序预设可自动判断部分设置值。若要自行设置，请使用--expert 选项。
- --kickstart：让程序自动检测并保存所有的鼠标设置。
- --noprobe：不要检测鼠标设备。
- --test：测试模式，不会改变任何设置。

应用实例如下。

使用交互界面设置鼠标：执行 mouseconfig 命令后会在命令行下弹出一个对话框，这时即可进行设定，如图 8-6 所示。

图 8-6　mouseconfig 配置界面

设定结束后按 Tab 键激活“确定”按钮并选择，即可保存设置并退出，然后使用命令 serxice network restart 激活即可生效。

8.79 bind：显示或设置键盘按键及其相关的功能

作用：可以用 bind 命令了解有哪些按键组合及其功能，也可以自行指定要用哪些按键组合。

用法：bind [-dlv] [-f <按键配置文件>] [-m <按键配置>] [-q <功能>]

主要选项如下。

– -d：显示按键配置的内容。
– -f<按键配置文件>：载入指定的按键配置文件。
– -l：列出所有的功能。
– -m<按键配置>：指定按键配置。
– -q<功能>：显示指定功能的按键。
– -v：列出目前的按键配置及其功能。

应用实例如下。

列出键盘按键所有的功能。

```
# bind -l
abort
accept-line
alias-expand-line
arrow-key-prefix
backward-byte
backward-char
...
```

8.80 kbdconfig：设置键盘类型

作用：kbdconfig（Red Hat Linux 才有的命令）是一个用来设置键盘的程序。kbdconfig 实际上是修改/etc/sysconfig/keyboard 的键盘配置文件，是 RHEL 老版本的工具。

用法：kbdconfig [--back] [--test]

主要选项如下。

– --back：执行时将预设的 Cancel 按钮更改为 Back 按钮。
– --test：仅做测试，不会实际更改设置。

8.81 snapscreenshot：命令行截图

作用：snapscreenshot 命令用来进行命令行屏幕截图。

用法：snapscreenshot [选项] > file

主要选项如下。

– -f, --fontwidth：设定字体大小，默认是 9。
– -b, --border：边框颜色。
– -1, --firstwin：第一个截图窗口。

– -c, --num：屏幕分割数目，默认是 6。
– -x, --alongside：屏幕水平分割数目，默认是 4。
– file：产生的文件名称，格式是 tga。

说明

snapscreenshot 可以获取控制台屏幕，并输出一个 TGA 格式的图像文件。如果想获得其他格式的图像文件，可以用图像转换程序（如 convert）将它转换成其他格式的文件。许多 Linux 发行版本中都包括这个工具，也可以通过 http://oktober.stc.cx/src/index/snapscreenshot/下载。

应用实例如下。

指定参数来获得图片，然后转换为 JPEG 格式，通过结合管道命令来实现。

```
#snapscreenshot|tgatopnm|pnmtojpeg>myscreen.jpeg
```

8.82 mt：磁带机控制

作用：mt 命令用来控制磁带机。

用法：mt[-v] [--version] [-h] [-f device]command[count]

主要选项如下。

– -v,-version：显示 mt—st 软件版本。
– -f：使用 Taper 设备。
– device：指明磁带设备。
– -h：显示 cmmand 子命令列表。
– count：前进或后退记录数目。

command 主要包括以下常用子命令。

– rewind：它可以使磁带快倒至磁带头。
– fsf：它可以使磁带快速前进若干段记录。
– bsf：它可以使磁带快速后退若干段记录。
– offline：使磁带脱机，即使磁带快倒至磁带头并弹出磁带机。
– status：报告设备状态。

由于 command 子命令比较多（共 34 个），详细的使用说明可参考 mt 手册页（Man Page）。

应用实例如下。

将 www 和 home 目录备份成两个 tar 文档到磁带机，使用如下命令：

```
#mt -f /dev/nst0 rewind
#tar cvf /dev/nst0 www；tar cvf /dev/ nst0 home
```

说明

由于不同系统所使用的磁带记录块大小不同，要想在 Linux 中读取 HP 工作站上写的磁带，要在读磁带前使用命令“mt -f 磁带机设备名 setblk 0”来设置可变记录块大小。磁带机不用时，可将其关闭，使用时再打开。只要主机在此期间不关闭，且没有使用操作磁带机的命令，磁带机设备就不会受到影响。

8.83 cdrecord：CD 刻录工具

作用：cdrecord 命令用来进行 CD 光盘刻录。

用法：cdrecord [选项] track1

主要选项如下。

- -scanbus：扫描 SCSI 刻录机设备号。
- -eject：刻录完成后自动弹出光盘。
- speed＝#：刻录速度。
- dev=target：刻录机设备号。
- -v：使用多段刻录模式。
- -V：使用 SCSI 多段刻录模式。

应用实例如下。

采用 Linux 命令行方式制作数据光盘：制作 CD-ROM 数据光盘时，通常分为以下几步。

①确认 Linux 主机目前使用的内核支持 SCSI 或 SCSI 仿真。

②如果目前使用 IDE 接口的光盘进行刻录，则要保证内核支持 Generic SCSI 驱动。

③安装 cdrecord 工具。

④扫描刻录机设备号。

⑤将文件刻录到光盘。

通常目前 2.4 以上内核的 Linux 发行版本都符合要求。

```
    # cdrecord -scanbus
    Cdrecord 2.0 (i686-pc-linux-gnu) Copyright (C) 1995-2002 Jörg Schilling
    Linux sg driver version: 3.1.24
    Using libscg version 'schily-0.7'
    cdrecord: Warning: using inofficial libscg transport code version
(schily-Red Hat-scsi-linux-sg.c-1.75-RH '@(#)scsi-linux-sg.c1.75 02/10/21
Copyright 1997 J. Schilling').
    scsibus0:
```

```
0,0,0   0)  'LG'   'CD-RW CED-8083B ' '1.05' Removable CD-ROM
0,1,0   1) *
0,2,0   2) *
0,3,0   3) *
0,4,0   4) *
0,5,0   5) *
0,6,0   6) *
0,7,0   7) *
```

可以看到扫描结果自动发现了刻录机的设备号 dev=0,0,0。下面刻录光盘，使用如下命令：

```
#cdrecord -v speed=24 dev=0,0,0 avg.bin
```

8.84 dvdrecord：DVD 刻录工具

作用：dvdrecord 命令用来进行 DVD 光盘刻录。

作用：dvdrecord [选项] track1

主要选项如下。

- -scanbus：扫描 SCSI 刻录机设备号。
- -eject：刻录完成后自动弹出光盘。
- speed＝#：刻录速度。
- dev=target：刻录机设备号。
- -v：使用多段刻录模式。
- -V：使用 SCSI 多段刻录模式。

> **说明**
>
> dvdrecord 是 cdrecord 在先锋 DVR-A30 平台上开发的 DVD 刻录升级工具，二者格式相同。

应用实例如下。

dvdrecord 是命令行下的 DVD 刻录软件，其最大优点是使用简单快速。如果要把 cajkodvd.img 刻录到 DVD 光盘，可使用如下命令：

```
#dvdrecord -scanbus
#dvdrecord -v speed＝2 dev＝0,1,0 -date /home/cao/cajkodvd.img
```

8.85 lspci：查看硬件插槽

作用：lspci 命令用来查看主板所有的硬件插槽信息。

用法：lspci [<switches>]

主要选项如下。

- -v：使得 lspci 以冗余模式显示所有设备的详细信息。
- -vv：使得 lspci 以过冗余模式显示更详细的信息（事实上是 PCI 设备能给出的所有信息）。这些数据的确切意义没有在此手册页中解释，如果用户想知道更多，请参考 /usr/include/linux/pci.h 或者 PCI 规范。
- -n：以数字形式显示 PCI 生产厂商和设备号，而不是在 PCI ID 数据库中查找它们。
- -x：以十六进制显示 PCI 配置空间（Configuration Space）的前 64 字节映象（标准头部信息）。此参数对调试驱动和 lspci 本身很有用。
- -xxx：以十六进制显示所有 PCI 配置空间的映象。此选项只有 root 可用，并且很多 PCI 设备在用户试图读取配置空间的未定义部分时会崩溃（此操作可能不违反 PCI 标准，但是它至少非常愚蠢）。
- -b：以总线为中心进行查看。显示所有 IRQ 号和内存地址。
- -t：以树形方式显示包含所有总线、桥、设备和它们的连接的图表。
- -s[[<bus>]:] [<slot>] [.[<func>]]：仅显示指定总线、插槽上的设备或设备上的功能块信息。设备地址的任何部分都可以忽略，或以“*”代替（意味着所有值）。所有数字都是十六进制。例如，“0:”指的是在 0 号总线上的所有设备；“0”指的是在任意总线上 0 号设备的所有功能块；“0.3”选择了所有总线上 0 号设备的第三个功能块；“.4”则是只列出每一个设备上的第四个功能块。
- -d[<vendor>]:[<device>]：只显示指定生产厂商和设备 ID 的设备。这两个 ID 都以十六进制表示，可以忽略或者以“*”代替（意味着所有值）。
- -i<file>：使用<file>作为 PCI ID 数据库而不是使用默认的/usr/share/hwdata/pci.ids。
- -p<dir>：使用<dir>作为包含 PCI 总线信息的目录而不是使用默认的目录/proc/bus/pci。
- -m：以机器可读的方式转储 PCI 设备数据（支持两种模式：普通和冗余），便于脚本解析。
- -M：使用总线映射模式，这种模式对总线进行全面扫描以查明总线上的所有设备，包括配置错误的桥之后的设备。请注意，此操作只应在调试时使用，并可能造成系统崩溃（只在设备有错误的时候，但不幸的是它们存在），此命令只有 root 可以使用。同时，在不直接接触硬件的 PCI 访问模式中使用-M 参数没有意义，因为显示的结果（排除 lspci 中的 bug 的影响）与普通的列表模式相同。

应用实例如下。

当系统硬件出现故障时，可以使用 lspci 命令输出所有硬件信息，以帮助确定故障点。

```
#lspci -m

00:00.0 "Host bridge" "Intel Corp." "440BX/ZX/DX - 82443BX/ZX/DX Host bridge"
```

```
-r03 "Asustek Computer, Inc." "8024"
    00:01.0 "PCI bridge" "Intel Corp." "440BX/ZX/DX - 82443BX/ZX/DX AGP bridge"
-r03
    00:04.0 "ISA bridge" "Intel Corp." "82371AB/EB/MB PIIX4 ISA" -r02 "" ""
    00:04.1 "IDE interface" "Intel Corp." "82371AB/EB/MB PIIX4 IDE" -r01 -p80 "" ""
    00:04.2 "USB Controller" "Intel Corp." "82371AB/EB/MB PIIX4 USB" -r01 "" ""
    00:04.3 "Bridge" "Intel Corp." "82371AB/EB/MB PIIX4 ACPI" -r02 "" ""
    00:09.0 "Communication controller" "Rockwell International" "HCF 56k PCI
Speakerphone Modem" -r08 "Rockwell International" "HCF 56k PCI Speakerphone
Modem"
    00:0a.0 "USB Controller" "OPTi Inc." "82C861" -r10 -p10 "OPTi Inc." "82C861"
    00:0b.0 "Ethernet controller" "Realtek Semiconductor Co., Ltd."
"RTL-8139/8139C/8139C+" -r10 "Realtek Semiconductor Co., Ltd." "RT8139"
    00:0d.0 "Ethernet controller" "Macronix, Inc. [MXIC]" "MX987x5" -r20 "" ""
    01:00.0 "VGA compatible controller" "nVidia Corporation" "NV5M64 [RIVA TNT2
Model 64/Model 64 Pro]" -r15 "" ""
```

可以看到 lspci 可以检测 ISA、PCI、AGP 扩展槽，硬盘通道，USB 总线等所有主板信息。

8.86 sane-find-scanner：扫描仪搜索

作用：sane-find-scanner 命令用来搜索本地的扫描仪接口。

用法：sane-find-scanner [-hvqf] [设备名称]

主要选项如下。

- -v：输出详细设备信息。
- -q：检测完毕自动退出。
- -f：强迫所有设备以 SCSI 模式运行。

> **说明**
>
> Linux 对 SANE 的支持包括两个方面：对底层接口设备的支持（SCSI 和 USB 等）；对一个具体型号的支持。使用扫描仪前要安装软件包 sane-frontends-1.0.9-2（实现 API 的库：中间件）、sane-backends-1.0.9-5（访问扫描仪的驱动程序：后端程序）、xsane-0.89-3（调用 API 的程序：前端程序）。

应用实例如下。

搜索本地扫描仪的接口，使用如下命令：

```
    sane-find-scanner
    # No SCSI scanners found. If you expected something different, make sure
that
    # you have loaded a SCSI driver for your SCSI adapter.
    found USB scanner (vendor=0x04a5, product=0x2060) at /dev/usb/scanner0
    found USB scanner (vendor=0x04a5 [Color], product=0x2060 [FlatbedScanner
13]) at libusb:002:002
    # A USB device was detected. This program can't be sure if it's really
    # a scanner. If it is your scanner, it may or may not be supported by
    # SANE. Try scanimage -L and read the backend's manpage.
    # Scanners connected to the parallel port or other proprietary ports can't be
    # detected by this program.
```

上面的结果表示没有发现 SCSI 接口扫描仪，检测到 USB 接口扫描仪。

8.87 scanimage：检测扫描仪型号

作用：scanimage 命令用来检测本地或者网络上扫描仪的具体型号。

用法：scanimage [选项]

主要选项如下。

- -d, --device-name=DEVICE：指定扫描仪型号（如 hp:/dev/scanner）。
- --format=pnm|tiff：指定输出文件格式（目前有.tiff 和.pnm 两种）。
- -L, --list-devices：列出能使用的扫描仪型号。
- -n, --dont-scan：模拟设置，不进行实际扫描。
- -T, --test：检测扫描仪后端程序。
- -h, --help：显示帮助信息。
- -v, --verbose：显示扫描仪工作状态。

应用实例如下。

（1）scanimage 加参数-L 用来检测扫描仪的具体型号。

```
# scanimage -L
device `snapscan:/dev/usb/scanner0' is a Acer FlatbedScanner13
```

可以看到扫描仪具体型号是：Acer FlatbedScanner13 平面扫描仪。

（2）在命令行下进行扫描。

```
# scanimage —d/dev/usb/scanner0 ——mode color >2.pnm
```

一个文件名是 2.pnm 的图片就生成了。

8.88 mtools：命令集

mtools 实际上是一个命令集合。mtools 是 DOS 文件系统的工具程序，它可以模拟许多 DOS 命令，使用起来非常方便。使用权限是所有用户。Linux 系统提供了一组名为 mtools 的可移植工具，可以让用户轻松地从标准的 DOS 软盘上读/写文件和目录。它们对 DOS 和 Linux 环境之间交换文件非常有用。它们是不具备共同的文件系统格式的系统之间交换文件的有力手段。一般 Linux 发行版本中都含有这个软件。可以使用如下命令检查一下：

```
#rpm –qa|grep mtools
```

如果没有安装，那也没有关系，可以从网上下载（http://mtools.linux.lu/）一个最新版本来安装。可供下载的最新 mtools 版本是 3.9.9。下载链接为 http://mtools.linux.lu/ mtools-3.9.9-3.i386.rpm，然后安装一下即可。

mtools 主要包括以下一些命令。

- mcd 目录名：改变 MS-DOS 下的目录。
- mcopy 源文件：目标文件在 MS-DOS 和 UNIX 之间复制文件。
- mdel 文件名：删除 MS-DOS 下的文件。
- mdir 目录名：显示 MS-DOS 下的目录。
- mformat 驱动器号：在低级格式化的软盘上创建 MS-DOS 文件系统。
- rnlabel 驱动器号：产生 MS-DOS 下的卷标。
- mmd 目录名：建立 MS-DOS 下的目录。
- mrd 目录名：删除 MS-DOS 下的目录。
- mren 源文件：目标文件重新命名已存在的 MS-DOS 文件。
- mtype 文件名：显示 MS-DOS 文件的内容。

提示

这些命令和对应的 MS-DOS 命令非常相似。需要说明的是，在 mtools 命令中，/和\是可以混用的。因为文件列表在 MS-DOS 系统下对大小写并不敏感，所以“CDE”和“cde”在这里是一样的。

应用实例如下。

（1）如果要把软盘进行快速格式化，可以使用命令 mformat。

```
mformat A:
```

当初发展 mtools 的目的是用来处理 DOS 文件系统，所以只能用在 FAT 文件格式的分区上。还需要注意的是，如果用 mount 命令挂载了 FAT16/32 分区，那么也不能使用 mtools 的命令来处理这些分区上的文件。这是因为一旦 FAT16/32 分区挂载到了 Linux 文件目录下，Linux 就会将其视为文件系统的一部分，这时要对其进行操作，就必须使用 Linux 本身所附带的命令集。

（2）将 MS-DOS 盘上的文件 htca.c 复制到当前目录下，并用 ls 命令进行验证。

```
$ mcopy a:\htca.c
$ ls –l htca.c
-rw-r– -r- - 1 xxq xxq 27136 Jan 1 01:80 htca.c
```

8.89 whereis：查找文件

作用：whereis 命令用于在特定目录中查找符合条件的文件（包括二进制文件、手册页文件、源码文件）。

用法： whereis [选项]文件

主要选项如下。

- -b：只搜索二进制文件。
- -B<目录>：定义二进制文件查找路径。
- -m：只搜索 man 手册。
- -M<目录>：定义 man 手册查找路径。
- -s：只搜索源代码。
- -S<目录>：定义源代码查找路径。
- -f：终止<目录>参数列表。
- -u：搜索不常见记录。
- -l：输出有效查找路径。

应用实例如下。

查看 ls 命令手册页位置。

```
# whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz /usr/share/man/man1p/ls.1p.gz
```

8.90 ytalk：与其他用户交谈

作用：通过 ytalk 命令，可以和其他用户线上交谈，如果想和其他主机的用户交谈，在用户名称后加上其主机名称或 IP 地址即可。

用法：ytalk [-isxY] [-h<主机名称 IP 地址>] [用户名称...]

主要选项如下。

- -h<主机名称 IP 地址>：指定交谈对象所在的远端主机。
- -i：用提醒声响代替显示信息。
- -s：在命令提示符后先开启 ytalk 交谈窗。

– -x：关闭图形界面。
– -Y：所有必须回应 yes 或 no 的问题，必须用大写英文字母 Y 或 N 回答。

应用实例如下。

```
# who           #显示当前用户
root   pts/1    Apr 9 20:17
tank   pts/6    May 27 16:47 (192.168.0.1)
root   pts/2    May 27 17:37 (192.168.0.1)
# ytalk tank    #发送消息
hey
```

8.91 apt：软件包在线管理

APT（Advanced Package Tool）是 Debian GNU/Linux 用来管理套件的工具，Conectiva 这家巴西的 Linux 套件公司将它移植到使用 RPM 系统的 Linux 套件上，所以现在我们才可以在 Red Hat Linux（或 Fedora）上使用 APT 来管理我们的 RPM 软件。apt 主要有 apt-cache、apt-cdrom、apt-config、apt-get 4 个命令，用户用得最多的应该就是 apt-get 命令。这些命令的详细说明请参见 man 手册，这里只对常用方法进行说明。

1．apt-get

– apt-get update：更新本地 apt-get 的本地数据库，使其与服务器的 pkglist 文件同步。在升级以前一般都要执行本命令实现与服务器的一致。
– apt-get check：验证本地系统的完整性。
– apt-get dist-upgrade：类似于 apt-get，但是将安装所有的基础软件包，并尽力升级一切软件包，在需要时安装新软件包。
– apt-get install package_name：安装某个软件包及其相关软件包。
– apt-get remove package_name：删除该软件包，同时删除依赖于它的软件包。
– apt-get source package_name：下载软件包的 source rpm。
– apt-get clean：删除保存在缓存目录下（/var/cache/apt/archives）的被下载的软件包。
– apt-get upgrade package_name：升级指定的软件包，并升级其依赖的软件包。

2．apt-cdrom

在安装系统以后，可以使用 apt-cdrom 命令把安装光盘加入到 apt 的 sources.list 中，用法如下：

```
# apt-cdrom add
```

3 次运行 apt-cdrom add 命令，用来加载 3 张安装光盘。在/etc/apt/sources.list 中应该包含如下内容。

```
rpm cdrom:[Gaga V1.1 disc3]/ RedHat gaga
rpm cdrom:[Gaga V1.1 disc2]/ RedHat gaga
```

```
rpm cdrom:[Gaga V1.1 disc1]/ RedHat gaga
```

这样以后就可以使用 apt-get 来安装以前未安装的安装光盘上的软件了。

若用户的服务器没有和互联网连接，那么修改/etc/apt/sources.list，使其包含下面的内容：

```
rpm cdrom:[Gaga V1.1 disc3]/ RedHat gaga
rpm cdrom:[Gaga V1.1 disc2]/ RedHat gaga
rpm cdrom:[Gaga V1.1 disc1]/ RedHat gaga
# apt-get update
```

3．apt-cache

- apt-cache showpkg：显示软件包的一些常规信息。
- apt-cache stats：显示相关的统计信息。
- apt-cache dump：显示缓存中的每个软件包的简要描述信息。
- apt-cache unmet：显示不符合一致性的依赖性关系。
- apt-cache show：显示指定软件包的记录信息。
- apt-cache search：查找软件包。
- apt-cache depends：显示软件包的依赖性关系。
- apt-cache pkgnames：列出所有的软件包。

4．apt-config

apt-config dump：显示目前的配置信息。

应用实例如下。

1）使用 apt 定时更新软件系统

编辑/etc/cron.daily/upgrade 文件，添加如下内容：

```
#!/bin/sh
apt-get update
apt-get -y dist-upgrade
apt-get clean
```

修改权限命令：

```
#chmod 755 /etc/cron.daily/upgrade
```

启动 crond：

```
chkconfig crond on
```

这样计算机就会每天定时更新软件了。

2）为 apt 设定代理

编辑 etc/apt/apt.conf 文件，在 proxy 选项字段栏目""; // http://user:pass@host:port/加入一行是以交通大学的代理服务器为范例：

```
Proxy http://proxy.nctu.edu.cn:3128 ; // http://user:pass@host:port/
```

3）让 apt 自动升级内核，不需要手动升级

编辑/etc/apt/apt.conf，找到如下一行：

```
Allow-Duplicated { "^kernel$"; "^kernel-"; "^kmodule-"; "^gpg-pubkey$" };
```

删掉粗黑体字部分即可。

8.92 yum：在线管理软件包

作用：yum 命令可以进行全面、便捷的软件包管理。

用法：yum <选项> <更新项目> <软件包名称>

主要选项如下。

- -h, --help：显示帮助信息并且退出。
- -t, --tolerant：允许错误连接。
- -c [config file]：设置配置文件。
- -R [minutes]：设置最大等待时间。
- -d [debug level]：设置调试等级。
- -e [error level]：设置错误等级。
- -y：如果 yum 在工作过程中需要使用者响应，这个参数可以直接回答 yes。
- --version：显示 yum 版本信息。
- --installroot=[path]：设置根目录。
- --enablerepo=[repo]：允许 yum 使用这个仓库。
- --disablerepo=[repo]：禁止 yum 使用这个仓库。
- -x [package], --exclude=[package]：要排除安装软件包。
- --noplugins：禁用 yum 插件。
- 更新项目：更新的动作包括 7 个部分。

1．列举包文件

（1）列出资源库中所有可以安装或更新的 rpm 包。

```
# yum list
```

（2）列出资源库中特定的可以安装或更新，以及已经安装的 rpm 包。

```
# yum list perl      //列出名为 perl 的包

# yum list perl*     //列出以 perl 开头的包
```

（3）列出资源库中所有可以更新的 rpm 包。

```
# yum list updates
```

（4）列出已经安装的所有的 rpm 包。

```
# yum list installed
```

（5）列出已经安装的但是不包含在资源库中的 rpm 包。

```
# yum list extras
```

注：extras 是 repos.d 中定义的资源列表名称。

2. 列举资源信息

（1）列出资源库中所有可以安装或更新的 rpm 包的信息。

```
# yum info
```

（2）列出资源库中特定的可以安装或更新的及已经安装的 rpm 包的信息。

```
# yum info perl      //列出 perl 包的信息
# yum info perl*     //列出以 perl 开头的所有包的信息
```

（3）列出资源库中所有可以更新的 rpm 包的信息。

```
# yum info updates
```

（4）列出已经安装的所有的 rpm 包的信息。

```
# yum info installed
```

（5）列出已经安装的但是不包含在资源库中的 rpm 包的信息。

```
# yum info extras
```

3. 搜索软件包

（1）搜索匹配特定字符的 rpm 包。

```
# yum search perl        //在包名称、包描述等中搜索
```

（2）搜索包含特定文件名的 rpm 包。

```
# yum provides realplay
```

4. 管理软件包

（1）安装、管理、删除 rpm 包。

```
# yum install perl  //安装 perl 包
# yum install perl* //安装以 perl 开头的包
```

（2）删除 rpm 包，包括与该包有依赖性的包。

```
# yum remove perl*  //删除 perl-* 所有包
```

5. 更新软件包

（1）检查可更新的 rpm 包。

```
# yum check-update
```

（2）更新所有的 rpm 包。

```
# yum update
```

（3）更新指定的 rpm 包，如更新 kernel 和 kernel-source。

```
# yum update kernel kernel-source
```

6. 清空缓存

（1）清除暂存中的 rpm 包文件。

```
# yum clean packages
```

（2）清除暂存中的 rpm 头文件。

```
# yum clearn headers
```

（3）清除暂存中旧的 rpm 头文件。

```
# yum clean oldheaders
```

（4）清除暂存中旧的 rpm 头文件和包文件。

```
# yum clearn all
```

7. 其他

（1）安装 Livna.org rpms GPG Key。

```
# rpm --import http://rpm.livna.org/RPM-LIVNA-GPG-KEY
```

（2）检查 GPG Key。

```
# rpm -qa gpg-pubkey*
```

（3）显示 Key 信息。

```
# rpm -qi gpg-pubkey-a109b1ec-3f6e28d5
```

（4）删除 Key。

```
# rpm -e gpg-pubkey-a109b1ec-3f6e28d5
```

Red Hat 和 Fedora 的软件安装命令是 rpm，但是用 rpm 安装软件最大的麻烦就是需要手动寻找安装该软件所需要的一系列依赖关系，非常麻烦不说，一旦软件不用了需要卸载的话，由于卸载掉了某个依赖关系而导致其他的软件不能用是非常恼人的。令人高兴的是，Fedora 终于推出了类似于 Ubuntu 中的 apt 的命令 yum，令 Fedora 的软件安装变得简单。yumd 的特点包括：可以同时配置多个资源库（repository）、简洁的配置文件（/etc/yum.conf）、自动解决增加或删除 rpm 包时遇到的依赖性问题、使用方便、保持与 rpm 数据库的一致性。

说明

yum 是 Yellow dog Updater Modified 的简称，起初是由 yellow dog 这一发行版的开发者 Terra Soft 研发的，用 python 写成，那时还叫作 yup（yellow dog updater），后经杜克大学的 Linux@Duke 开发团队进行改进，遂有此名。yum 的宗旨是自动化地升级、安装/移除 rpm 包、收集 rpm 包的相关信息、检查依赖性并自动提示用户解决。yum 的关键之处是要有可靠的 repository，顾名思义，这是软件的仓库，它可以是 HTTP 或 FTP 站点，也可以是本地软件池，但必须包含 rpm 的 header。header 包括了 rpm 包的各种信息，包括描述、功能、提供的文件、依赖性等。正是收集了这些 header 并加以分析，才能自动化地完成余下的任务。

下面了解一下 yum.conf。

yum 的一切配置信息都存储在一个名为 yum.conf 的配置文件中，该文件通常位于/etc 目录下。

```
#cat /etc/yum.conf
[main]
cachedir=/var/cache/yum
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
metadata_expire=1800
# PUT YOUR REPOS HERE OR IN separate files named file.repo
# in /etc/yum.repos.d
```

下面对这一文件做简要的说明。

- cachedir：yum 缓存的目录，yum 在此存储下载的 rpm 包和数据库，一般是/var/cache/yum。
- debuglevel：除错级别，范围是 0～10，默认是 2。
- logfile：yum 的日志文件，默认是/var/log/yum.log。
- exactarch：有 1 和 0 两个选项，代表是否只升级和用户安装的软件包 CPU 体系一致的包。如果设为 1，那么如果用户安装了一个 i386 的 rpm，则 yum 不会用 686 的包来升级。
- gpgchkeck=：有 1 和 0 两个选项，分别代表是否进行 GPG 校验。

应用实例如下。

（1）安装所有和 MYSQL 有关的软件包。

```
#um install *mysql*
```

（2）检查是否有更新的软件包。

```
#um check-update
```

（3）检查所有提供 PHP 的软件包，并显示简介信息。

```
#um search php
```

（4）显示已经安装的软件包。

```
#um list installed
```

（5）移除 wget 软件包。

```
#um remove wget
```

8.93 vlock：锁定终端

作用：vlock 命令用来锁定终端。

用法：vlock [-achv]

主要选项如下。

- -a,--all：锁定所有的主控制台阶段操作；如果在主控制台执行该参数，则会关闭键盘切换终端的功能。
- -c,--current：锁定当前的主控制台阶段操作。

应用实例如下。

在单独执行 vlock 命令的情况下会锁定当前使用的控制台阶段操作，用户需要输入密码才能解除锁定，在命令行提示符下输入：

```
#vlock
```

8.94 wait：等待程序返回状态

作用：wait 命令用于等待程序返回状态。

用法：wait [编号]

应用实例如下。

等待进程 ID 为 1923 的程序的返回值，在命令行提示符下输入：

```
wait 1923
```

8.95 watch：将结果输出到标准输出设备

作用：watch 命令用于将结果输出到标准输出设备。

用法：watch[-dhntv] [--differences[=cumulative]] [--help] [--interval=<n>] [--no-title] [--version] <command>

主要选项如下。

- -d,--differences：显示差异。
- -n <seconds> --interval=<seconds>：周期性执行命令的间隔。
- -t,--no-title：关闭标题栏及下面的空白行。
- command：要周期性运行的程序。
- -t,--no-title：关闭 header。

应用实例如下。

（1）设置周期性执行命令的间隔，例如每隔 10 秒执行一次 ls 命令。

```
# watch -n 10 ls
```

（2）高亮显示 ls -l。

```
# watch -d ls -l
```

8.96 rsync：远程数据同步工具

作用：rsync（Remote Sync）是 UNIX/Linux 系统下一款优秀的数据备份与同步工具，它可以对文件集进行同步。然而更有价值的是，rsync 使用文件的增量，也就是说，它在网络中仅发送两个文件集合有区别的部分。这样可以占用更少的带宽，并且速度更快。

用法：rsync [选项]

主要选项如下。

- -a：该参数表示以递归方式传输文件，并保持所有文件属性。
- -v：该参数显示详细的传输处理。
- -z：该参数对备份的文件在传输时进行压缩处理。
- -P：该参数将显示备份过程。
- --delete：从目标目录中删除源目录中不再存在的文件。默认模式仅仅同步文件修改并创建新文件。在默认情况下，如果在源目录中删除了一个文件，就会忽略它，并不在目标目录中相应地删除它。通过使用这个选项，可以创建完全相同的同步。
- --recursive：递归地复制目录和文件。
- --times：同步每个文件和目录的修改时间及创建时间。
- --owner：如果可能的话，保留文件的所有者。
- --group：如果可能的话，保留组所有者。

- --links：把符号链接复制为符号链接，而不是复制文件数据并解释源链接。
- --perms：保留文件权限。
- --hard-links：保留硬链接（在目标目录中创建硬链接），而不是复制文件内容。

应用实例如下。

下面是一个备份远程 Web 页面的脚本。

```
#!/bin/sh
cd /backup
/usr/bin/rsync -avP --delete -l -t -e
ssh user@remote.com:~/public_html
```

这个脚本改变目录到/backup 目录，然后激活 rsync 复制远程主机 remote.com 上 user 用户的～/public_html 目录。这个脚本把所有的数据通过 ssh 加密，删除那些没有存在于远程主机的本地数据。备份的结果是将远程的～/public_html 目录复制到本地的/backup/public_ html。

> **说明**
>
> 谨慎使用--delete 选项，最好不要把已经有重要数据的目录当作客户端的备份目录，否则会误删除重要数据。

8.97 &：将任务放在后台执行

作用：将任务放在后台执行。

用法：command &

应用实例如下。

将命令或任务放到后台执行。例如在后台运行 find 程序，在命令行提示符下输入：

```
#find &
```

8.98 screen：多重视窗管理程序

作用：screen 为多重视窗管理程序。此处所谓的视窗，是指一个全屏幕的文字模式画面。简单来说，screen 是一个可以在多个进程之间多路复用一个物理终端的窗口管理器。

用法：screen [选项]

screen [-opts] [cmd [args]]

screen -r [host.tty]

主要选项如下。

- -A：将所有的视窗都调整为目前终端机的大小。
- -d<作业名称>：将指定的 screen 作业离线。
- -h<行数>：指定视窗的缓冲区行数。
- -m：即使目前已在作业中的 screen 作业，仍强制建立新的 screen 作业。
- -r<作业名称>：恢复离线的 screen 作业。
- -R：先试图恢复离线的作业。若找不到离线的作业，即建立新的 screen 作业。
- -s<shell>：指定建立新视窗时所要执行的 Shell。
- -S<作业名称>：指定 screen 作业的名称。
- -v：显示版本信息。
- -x：恢复之前离线的 screen 作业。
- -ls 或--list：显示目前所有的 screen 作业。
- -wipe：检查目前所有的 screen 作业，并删除已经无法使用的 screen 作业。
- -c <文件>：从文件$HOME/.screenrc 中重新将默认配置文件写入指定文件中。

在默认情况下，screen 接收以 C-a 开始的命令。这种命令形式在 screen 中叫作键绑定（Key Binding），C-a 叫作命令字符（Command Character）。可以通过“C-a ?”来查看所有的键绑定。常用的键绑定如下。

- C-a ?：显示所有的键绑定信息。
- C-a w：显示所有窗口列表。
- C-a C-a：切换到之前显示的窗口。
- C-a c：创建一个新的运行 Shell 的窗口并切换到该窗口。
- C-a n：切换到下一个窗口。
- C-a p：切换到前一个窗口（与 C-a n 相对）。
- C-a 0..9：切换到窗口 0..9。
- C-a a：发送 C-a 到当前窗口。
- C-a d：暂时断开 screen 会话。
- C-a k：杀掉当前窗口。
- C-a [：进入复制/回滚模式。

说明

Screen 可以看作窗口管理器的命令行界面版本，它提供了统一的管理多个会话的界面和相应的功能。

只要 Screen 本身没有终止，在其内部运行的会话都可以恢复。这一点对于远程登录的用户特别有用——即使网络连接中断，用户也不会失去对已经打开的命令行会话的控制。只要再次登录到主机上执行 screen -r 就可以恢复会话的运行。同样，在暂时离开的时候，也可以执行分离命令 detach，在保证里面的程序正常运行的情况下让 Screen 挂起（切换到后台）。这一点和

图形界面下的 VNC 很相似。

在 Screen 环境下，所有的会话都独立运行，并拥有各自的编号、输入、输出和窗口缓存。用户可以通过快捷键在不同的窗口下切换，并可以自由地重定向各个窗口的输入和输出。Screen 实现了基本的文本操作，如复制、粘贴等；还提供了类似滚动条的功能，可以查看窗口状况的历史记录。窗口可以被分区和命名，还可以监视后台窗口的活动。

Screen 可以让一个或多个用户从不同终端多次登录一个会话，并共享会话的所有特性（如可以看到完全相同的输出）。它同时提供了窗口访问权限的机制，可以对窗口进行密码保护。

应用实例如下。

（1）显示当前所有的 screen 操作。

```
# screen -ls

There is a screen on:

5130.pts-1.localhost(Attached)

1 Socket in /var/run/screen/S-root.
```

（2）将连接中的 screen 离线。

```
screen -d
```

（3）建立新的 screen 操作，并指定操作名称为 screen1。

```
#screen -S screen1
```

8.99 lsb_release：显示 LSB 和特定版本的相关信息

作用：LSB 是 Linux Standard Base 的缩写，lsb_release 命令用来显示 LSB 和特定版本的相关信息。

用法：lsb_release [选项]..

主要选项如下。

- -i,--id：显示发行版的 ID。
- -d,--description：显示该发行版的描述信息。
- -r,--release：显示当前系统是发行版的具体版本号。
- -c,--codename：发行版代号。
- -a,--all：显示上面的所有信息。
- -s,--short：输出简短的描述信息。

应用实例如下。

查看 Linux 是什么版本。

```
# lsb_release -a
LSB
```

```
Version: :core-4.1-amd64:core-4.1-noarch:cxx-4.1-amd64:cxx-4.1-noarch:deskt
op-4.1-amd64:desktop-4.1-noarch:languages-4.1-amd64:languages-4.1-noarch:pr
inting-4.1-amd64:printing-4.1-noarch
    Distributor ID:     CentOS
    Description:        CentOS Linux release 7.0.1406 (Core)
    Release:            7.0.1406
    Codename:           Core
```

8.100 lscpu：查看 CPU 信息

作用：查看 CPU 信息。

用法：lscpu [选项]

主要选项如下。

– -a, --all：同时打印在线和离线 CPU（-e 选项默认值）。
– -b, --online：只打印在线 CPU（-p 选项默认值）。
– -c, --offline：只打印离线 CPU。
– -e, --extended[=<列表>]：打印扩展的可读格式。
– -p, --parse[=<列表>]：打印可解析格式。
– -s, --sysroot <目录>：以指定目录作为系统根目录。
– -x, --hex：打印十六进制掩码而非 CPU 列表。

可显示的列如下。

– CPU：逻辑 CPU 数量。
– CORE：逻辑核心数量。
– SOCKET：逻辑（CPU）座数量。
– NODE：逻辑 NUMA 节点数量。
– BOOK：逻辑 book 数。
– CACHECPU：共享缓存。
– ADDRESS：物理地址。

应用实例如下。

查看当前 CPU 信息。

```
    # lscpu
    Architecture:                      x86_64
    CPU op-mode(s):                    32-bit, 64-bit
    Byte Order:                        Little Endian
    CPU(s):                            1
    On-line CPU(s) list:               0
```

```
Thread(s) per core:                 1
Core(s) per socket:                 1
座：                                1
NUMA 节点：                         1
厂商 ID：                           GenuineIntel
CPU 系列：                          6
型号：                              23
型号名称：                          Intel(R) Celeron(R) CPUE3200@ 2.40GHz
步进：                              10
CPU MHz：                           0.000
BogoMIPS：                          46333.95
L1d 缓存：                          32K
L1d 缓存：                          32K
L2d 缓存：                          6144K
NUMA 节点 0 CPU：                   0
```

8.101 blkid：查看块设备

作用：查看块设备（包括交换分区）的文件系统类型、LABEL、UUID、挂载目录等信息。

用法：blkid [选项] [<设备>...]

主要选项如下。

- -c <file>：指定 cache 文件。
- -h：显示帮助信息。
- -o <format>：指定输出格式。
- -s <tag>：显示指定信息，默认显示所有信息。
- -v：显示版本信息。

应用实例如下。

（1）列出当前系统中所有已挂载文件系统的类型。

```
# blkid
```

（2）显示指定设备 UUID。

```
# blkid -s UUID /dev/sda5
```

（3）显示所有设备 UUID。

```
# blkid -s UUID
```

（4）显示指定设备 LABEL。

```
# blkid -s LABEL /dev/sda5
```

（5）显示所有设备 LABEL。

```
# blkid -s LABEL
```

（6）显示所有设备文件系统。

```
# blkid -s TYPE
```

（7）显示所有设备。

```
# blkid -o device
```

（8）以列表方式查看详细信息。

```
# blkid -o list
```

8.102 journalctl：日志管理

作用：journal 是 systemd 的一个组成部分，负责查看和管理日志文件，它可以用于代替传统的 syslog 守护进程，包括 rsyslogd。

用法：journalctl [选项]

主要选项如下。

- --system：仅显示系统日志。
- --user：只显示当前用户的用户日志。
- --since=DATE：开始显示比指定日期新的日志。
- --until=DATE：停止显示比指定日期旧的日志。
- -b—boot：显示引导开始的日志。
- -k—dmesg：从当前引导开始显示内核消息日志。
- -u --unit=UNIT：仅显示指定的单元中的数据。
- --user-unit=UNIT：只能从指定的用户会话单元中显示数据。
- -p --priority=RANGE：显示在指定的优先级范围内唯一的数据。
- -f—follow：实时输出日志。
- -n --lines[=INTEGER]：显示指定行数的日志。
- --no-tail：显示所有行。
- -r—reverse：最先显示最新的条目。
- -o--output=STRING：更改日志输出模式（包括短、精确、短单调、冗长等模式）。
- -x--catalog：添加信息的解释。
- -a—all：显示日志行的所有字段。
- -q—quiet：不显示错误和报警信息。
- -D--directory：显示指定目录下的日志文件。
- --file：显示日志文件路径。

主要子命令如下。

- -h--help：显示此帮助文本。
- --version：显示软件包版本。
- --new-id128：生成一个新的 128 位 ID。
- --header：显示日志表头信息。
- --disk-usage：显示所有的日志文件的磁盘使用率情况。
- --verify：检查日志文件的内部一致性。
- --verify-key：指定 FSS 验证密钥以用于验证操作。

使用方法如下。

开启新日志系统非常简单，只需建个目录然后重启就行了。

```
$ sudo mkdir -p /var/log/journal
$ reboot
```

然后使用 su 命令进入超级用户，把自己添加到 adm 组，这样自己才能看到所有的日志内容。

```
# usermod -a -G adm cjh
```

然后重新登录就能通过 journalctl 看到所有的日志。

启动服务，使用如下命令。

```
#systemctl restart systemd-journald
```

常用命令实例如下。

（1）查看特定命令的日志。

```
$ journalctl _COMM=sshd
```

输出界面如图 8-7 所示。

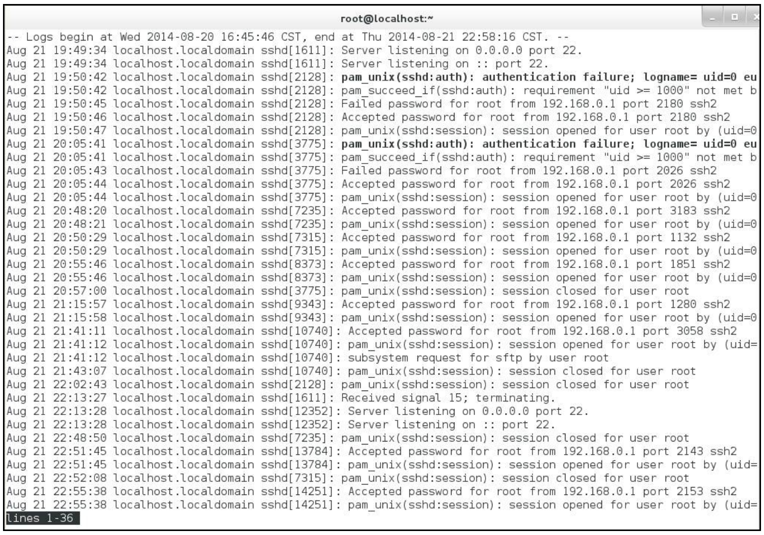


图 8-7　sshd 的输出界面

（2）查看 sshd 命令更详细的情况。

```
$ journalctl /usr/sbin/sshd -o verbose
```

（3）查看日志的 30 行。

```
$ journalctl -n 30
```

（4）查看特定命令在指定时间内的日志。

```
# journalctl _COMM=sudo --since "10:00" --until "11:00"
```

（5）设置过滤条件，只显示错误信息。

```
# journalctl -p err
```

输出界面如图 8-8 所示。

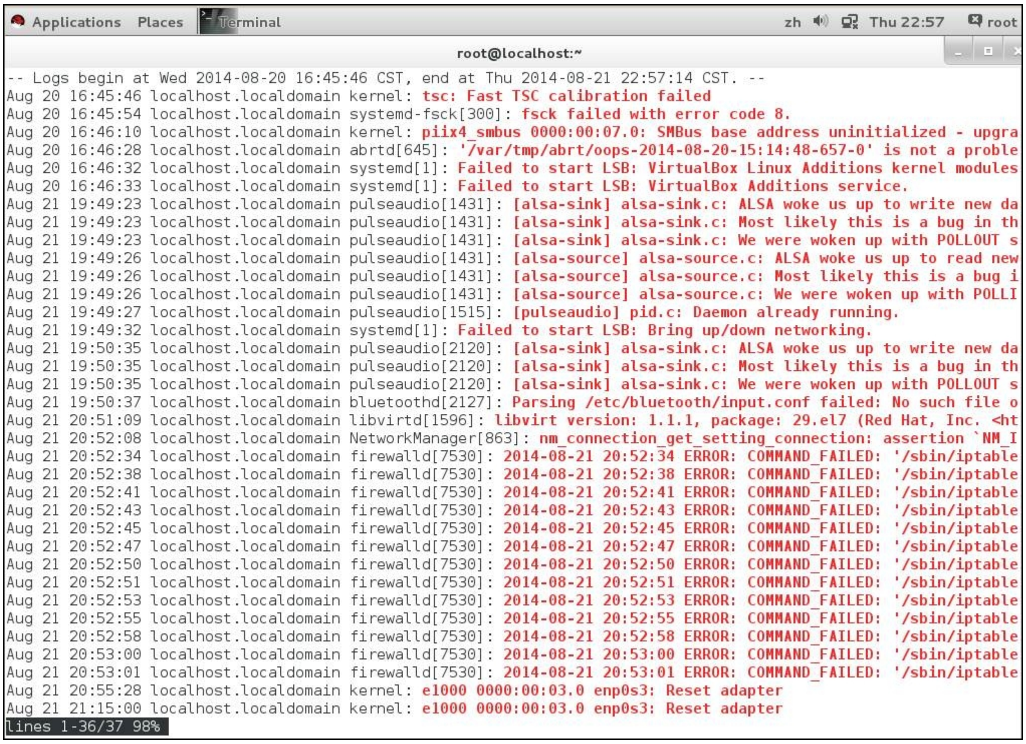


图 8-8　只显示错误信息

> **说明**
>
> 加粗字体说明出现故障。

（6）动态跟踪最新信息。

```
# journalctl -f
```

（7）显示特定程序的所有消息。

```
# journalctl /usr/lib/systemd/systemd
```

（8）显示特定进程的所有消息。

```
# journalctl _PID=1
```

（9）显示指定单元的所有消息。

```
# journalctl -u netcfg
```

（10）显示优先级为 2 的日志。

```
#journalctl -p 2 --since=today
```

这包括 emerg（最高的紧急程度状态）、alert（紧急状态）和 crit（重要信息）；酷似 syslog 优先级 emerg（0）、alert（1）、crit（2）、err（3）、warning（4）、notice（5）、info（6）和 debug（7）。

（11）读取最后一次启动以来的 boot 日志。

```
#journalctl --boot
```

（12）读取某一个服务的日志。

```
#journalctl -u apache.service
```

（13）显示指定时间段的日志。

- 查看昨晚登录的 httpd 进程。

```
#journalctl -u httpd since=00:00 --until=8:00
```

- 只显示自从昨天登录开始的信息。

```
#journalctl --since=yesterday
```

8.103 lsblk：列出块设备

作用：列出块设备。除了 RAM 外，以标准的树状输出格式整齐地显示块设备。lsblk 是用来了解新插入的 USB 设备名字的最有用和最简单的方式，特别是当用户在终端上处理磁盘/块设备时。

用法：lsblk [选项] [<设备> ...]

主要选项如下。

- -a,--all：打印所有设备。
- -b,--bytes：以字节为单位而非易读的格式来打印 SIZE。
- -d,--nodeps：不打印从属设备（slave）或占位设备（holder）。
- -D,--discard：打印时丢弃能力。

- -e,--exclude<列表>：根据主设备号排除设备（默认：内存盘）。
- -I,--include<列表>：只显示有指定主设备号的设备。
- -f,--fs：输出文件系统信息。
- -h,--help：使用信息（此信息）。
- -i,--ascii：只使用 ASCII 字符。
- -m,--perms：输出权限信息。
- -l,--list：使用列表格式的输出。
- -n,--noheadings：不打印标题。
- -o,--output<列表>：输出列。
- -p,--paths：打印完整设备路径。
- -P,--pairs：使用 key="value"输出格式。
- -r,--raw：使用原生输出格式。
- -s,--inverse：反向依赖。
- -t,--topology：输出拓扑信息。
- -S,--scsi：输出有关 SCSI 设备的信息。

应用实例如下。

以列表格式显示块设备（而不是树状格式）。

```
# lsblk -l
NAME MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda    8:0    0 232.9G  0 disk
sda1   8:1    0  46.6G  0 part /
sda2   8:2    0     1K  0 part
sda5   8:5    0   190M  0 part /boot
sda6   8:6    0   3.7G  0 part [SWAP]
sr0   11:0    1  1024M  0 rom
```

8.104 systemd：服务管理命令组

systemd 的主要命令行工具是 systemctl。多数 Linux 系统管理员应该已经非常熟悉系统服务和 init 系统的管理，比如 service、chkconfig 及 telinit 命令的使用。systemd 也完成同样的管理任务，只是命令工具 systemctl 的语法有所不同而已，因此用表格来对比 systemctl 和传统的系统管理命令会非常清晰。

1．SysVinit 和 Systemd 的命令对比列表

服务命令对比如表 8-6 所示，这里以 sshd 服务为例。表 8-6 的目的在于帮助系统管理员了解 Systemd 中可以取代原先 SysVinit 工作流程的命令。注意，service 和 chkconfig 这两条命令在 Systemd 环境下依然可用。

表 8-6　SysVinit 和 Systemd 服务命令对比

Sysvinit 命令	Systemd 命令	备　注
service sshd start	systemctl start sshd.service	用来启动一个服务（并不会重启现有的）
service sshd stop	systemctl stop sshd.service	用来停止一个服务（并不会重启现有的）
service sshd restart	systemctl restart sshd.service	用来停止并启动一个服务
service sshd reload	systemctl reload sshd.service	当支持时，重新装载配置文件而不中断等待操作
service sshd condrestart	systemctl condrestart sshd.service	如果服务正在运行，那么重启它
service sshd status	systemctl status sshd.service	汇报服务是否正在运行
ls /etc/rc.d/init.d/	systemctl list-unit-files --type=service ls /lib/systemd/system/*.service etc/systemd/system/*.service	用来列出可以启动或停止的服务列表
chkconfig sshd on	systemctl enable sshd.service	在下次启动时或满足其他触发条件时设置服务为启用
chkconfig sshd off	systemctl disable sshd.service	在下次启动时或满足其他触发条件时设置服务为禁用
chkconfig sshd	systemctl is-enabled sshd.service	用来检查一个服务在当前环境下被配置为启用还是禁用
chkconfig --list	systemctl list-unit-files --type=service（推荐） ls /etc/systemd/system/*.wants/	输出在各个运行级别下服务的启用和禁用情况
chkconfig sshd --list	ls /etc/systemd/system/*.wants/sshd.service	用来列出该服务在哪些运行级别下启用和禁用
chkconfig sshd --add	systemctl daemon-reload	在创建新服务文件或者变更设置时使用

注意

以上列出的所有/sbin/service 和/sbin/chkconfig 在 Systemd 环境下依然可以工作，并且在必要的时候将会被翻译成原生的等效命令。唯一的例外是 chkconfig --list。

改变运行级别对比如表 8-7 所示。

表 8-7　SysVinit 和 Systemd 改变运行级别对比

Sysvinit 命令	Systemd 命令	备　注
telinit 3	systemctl isolate multi-user.target 或者 systemctl isolate runlevel3.target OR telinit 3	改变至多用户运行级别
sed s/^id:.*:initdefault:/id:3:initdefault:/	ln -sf /lib/systemd/system/multi-user.target /etc/systemd/system/default.target	设置在下一次启动时使用多用户运行级别

电源管理命令对比如表 8-8 所示。

表 8-8　SysVinit 和 Systemd 电源管理命令对比

Systemd 命令	Sysvinit 命令	操　作
systemctl reboot	reboot	重启机器
systemctl poweroff	halt -p	关机
systemctl suspend	echo standby > /sys/power/state	待机
systemctl hibernate	echo platform > /sys/power/disk; echo disk >/sys/power/state 或者 echo shutdown > /sys/power/disk; echo disk >/sys/power/state	休眠
systemctl hybrid-sleep		混合休眠模式（同时休眠到硬盘并待机）

关机不是每个登录用户在任何情况下都可以执行的，一般只有管理员才可以关机。

2．systemctl 命令实例

1）systemctl 的基本用法

- systemctl：列出所有已加载的单元及其状态（单元是任务/服务的术语）。
- systemctl list-units：列出所有单元。
- systemctl start [NAME...]：启动（激活）一个或多个单元。
- systemctl stop [NAME...]：停止（停用）一个或多个单元。
- systemctl disable [NAME...]：禁用一个或多个单元文件。
- systemctl list-unit-files：显示所有已安装的单元文件及其状态。
- systemctl --failed：显示启动过程中哪些单元出现故障。
- systemctl --type=mount：类型过滤器；类型可能是服务、挂载点、设备、套接字和启动目标。
- systemctl enable debug-shell.service：启动 TTY 9 上的 Shell，用于调试。

如果想更方便地处理单元，以及程序包 systemd-ui，它可以由用户使用 systemadm 命令来启动。

切换运行级别、重启和关闭系统也由 systemctl 来处理。

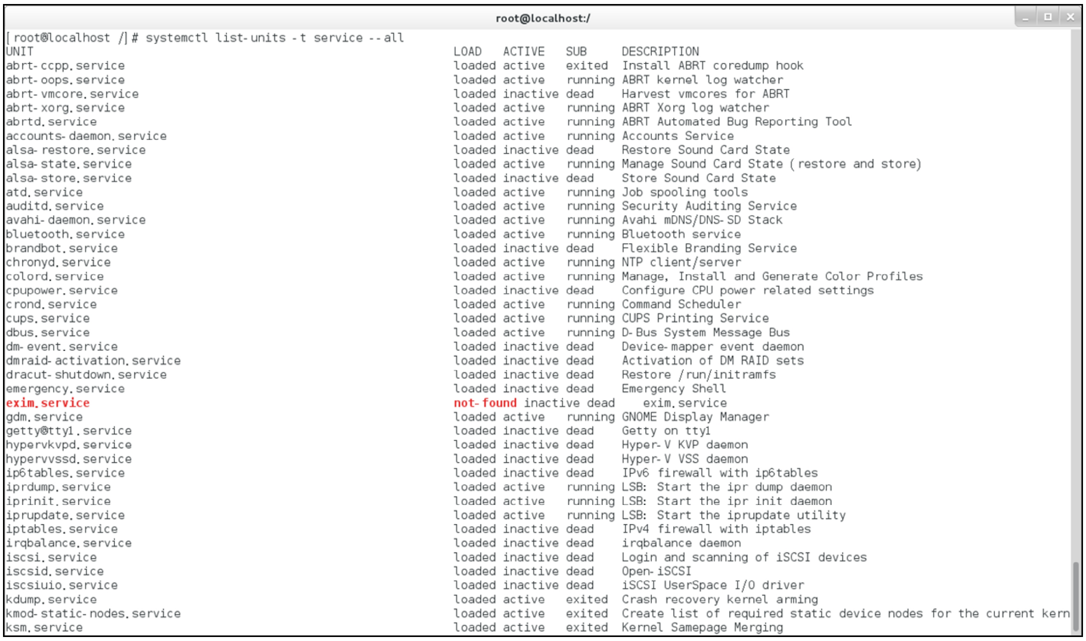
- systemctl isolate graphical.target：切换到运行级别 5，X 服务器在运行级别 5 运行。
- systemctl isolate multi-user.target：切换到运行级别 3 和 TTY，不带 X 图形界面。
- systemctl reboot：关闭和重启系统。
- systemctl poweroff：关闭系统。

除了用于切换运行级别的命令外，所有这些命令由普通用户的权限就可以执行。

（1）显示所有已经安装的服务。

```
#systemctllist-unit-files -t service –all
```

命令输出界面如图 8-9 所示。



```
root@localhost:/
[root@localhost /]# systemctl list-units -t service --all
UNIT                                  LOAD      ACTIVE   SUB     DESCRIPTION
abrt-ccpp.service                     loaded    active   exited  Install ABRT coredump hook
abrt-oops.service                     loaded    active   running ABRT kernel log watcher
abrt-vmcore.service                   loaded    inactive dead    Harvest vmcores for ABRT
abrt-xorg.service                     loaded    active   running ABRT Xorg log watcher
abrtd.service                         loaded    active   running ABRT Automated Bug Reporting Tool
accounts-daemon.service               loaded    active   running Accounts Service
alsa-restore.service                  loaded    inactive dead    Restore Sound Card State
alsa-state.service                    loaded    active   running Manage Sound Card State (restore and store)
alsa-store.service                    loaded    inactive dead    Store Sound Card State
atd.service                           loaded    active   running Job spooling tools
auditd.service                        loaded    active   running Security Auditing Service
avahi-daemon.service                  loaded    active   running Avahi mDNS/DNS-SD Stack
bluetooth.service                     loaded    active   running Bluetooth service
brandbot.service                      loaded    inactive dead    Flexible Branding Service
chronyd.service                       loaded    active   running NTP client/server
colord.service                        loaded    active   running Manage, Install and Generate Color Profiles
cpupower.service                      loaded    inactive dead    Configure CPU power related settings
crond.service                         loaded    active   running Command Scheduler
cups.service                          loaded    active   running CUPS Printing Service
dbus.service                          loaded    active   running D-Bus System Message Bus
dm-event.service                      loaded    inactive dead    Device-mapper event daemon
dmraid-activation.service             loaded    inactive dead    Activation of DM RAID sets
dracut-shutdown.service               loaded    inactive dead    Restore /run/initramfs
emergency.service                     loaded    inactive dead    Emergency Shell
exim.service                          not-found inactive dead    exim.service
gdm.service                           loaded    active   running GNOME Display Manager
getty@tty1.service                    loaded    inactive dead    Getty on tty1
hypervkvpd.service                    loaded    inactive dead    Hyper-V KVP daemon
hypervvssd.service                    loaded    inactive dead    Hyper-V VSS daemon
ip6tables.service                     loaded    inactive dead    IPv6 firewall with ip6tables
iprdump.service                       loaded    active   running LSB: Start the ipr dump daemon
iprinit.service                       loaded    active   running LSB: Start the ipr init daemon
iprupdate.service                     loaded    active   running LSB: Start the iprupdate utility
iptables.service                      loaded    inactive dead    IPv4 firewall with iptables
irqbalance.service                    loaded    inactive dead    irqbalance daemon
iscsi.service                         loaded    inactive dead    Login and scanning of iSCSI devices
iscsid.service                        loaded    inactive dead    Open-iSCSI
iscsiuio.service                      loaded    inactive dead    iSCSI UserSpace I/O driver
kdump.service                         loaded    active   exited  Crash recovery kernel arming
kmod-static-nodes.service             loaded    active   exited  Create list of required static device nodes for the current kern
ksm.service                           loaded    active   exited  Kernel Samepage Merging
```

图 8-9　显示所有服务

如图 8-9 所示的输出界面包括 5 列。

– UNIT：显示的是服务的名称。
– LOAD：显示该服务是否已经加载，如果没有发现则显示为红色。
– ACTIVE：显示高级单元是否已经激活，如果没有激活则显示为红色。
– SUB：显示低级单元是否已经激活，如果没有激活则显示为红色。
– DESCRIPTION：显示的是服务的功能说明。

（2）显示 HTTPD 服务的详细信息。

```
systemctl status httpd.service
```

2）管理服务

启动一个服务：systemctl enable service_name.service。

例如：systemctl enable httpd.servic。

停止一个服务：systemctl disable service_name.service。

例如：systemctl disable httpd.service。

（1）显示一个服务的相互依存情况。

```
# systemctl list-dependencies nfs-server.service
```

（2）在开机时启用一个服务。

```
#systemctl enable foo.service
```

（3）在开机时禁用一个服务。

```
#systemctl disable foo.service
```

（4）检查一个服务是否已开机启用。

```
#systemctl is-enabled foo.service; echo $?
```

0 表示已开机启用，1 表示没有开机启用。在 Fedora 17 中，除了返回值外，相应的“enable”或“disable”也会显示到标准输出。

（5）修改了 httpd.service 后，通过以下命令就能让系统使用新的设置。

```
#systemctl daemon-reload
#systemctl restart httpd.service
```

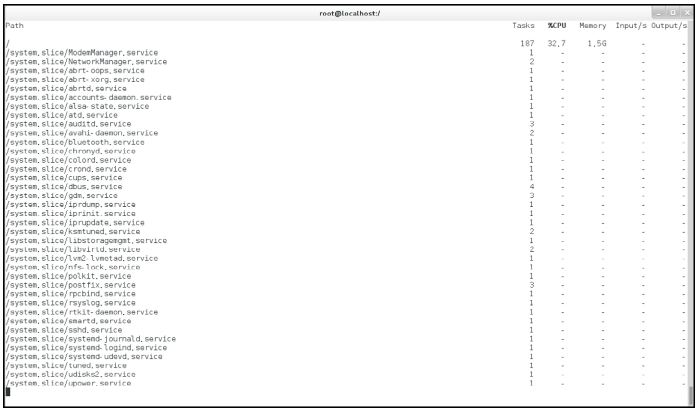
（6）检查一个服务的运行情况。

```
# systemctl status crond.service
crond.service - Command Scheduler
Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled)
Active: active (running) since Mon, 02 Jul 2014 18:43:54 +0900; 42min ago
Main PID: 3500 (crond)
CGroup: name=systemd:/system/crond.service
 └ 3500 /usr/sbin/crond -n
Jul 02 18:43:55 dragon.nrt.redhat.com /usr/sbin/crond[3500]: (CRON) INFO
(running with inotify support)
Jul 02 18:43:55 dragon.nrt.redhat.com /usr/sbin/crond[3500]: (CRON) INFO
(@reboot jobs will be run at computer's startup.)
```

（7）检查资源使用情况，包括 CPU、内存、I/O。

```
# systemd-cgtop
```

检查结果如图 8-10 所示。



root@localhost:/

Path	Tasks	%CPU	Memory	Input/s	Output/s
/	187	32.7	1.5G	-	-
/system.slice/ModemManager.service	1	-	-	-	-
/system.slice/NetworkManager.service	2	-	-	-	-
/system.slice/abrt-oops.service	1	-	-	-	-
/system.slice/abrt-xorg.service	1	-	-	-	-
/system.slice/abrtd.service	1	-	-	-	-
/system.slice/accounts-daemon.service	1	-	-	-	-
/system.slice/alsa-state.service	1	-	-	-	-
/system.slice/atd.service	1	-	-	-	-
/system.slice/auditd.service	3	-	-	-	-
/system.slice/avahi-daemon.service	2	-	-	-	-
/system.slice/bluetooth.service	1	-	-	-	-
/system.slice/chronyd.service	1	-	-	-	-
/system.slice/colord.service	1	-	-	-	-
/system.slice/crond.service	1	-	-	-	-
/system.slice/cups.service	1	-	-	-	-
/system.slice/dbus.service	4	-	-	-	-
/system.slice/gdm.service	3	-	-	-	-
/system.slice/iprdump.service	1	-	-	-	-
/system.slice/iprinit.service	1	-	-	-	-
/system.slice/iprupdate.service	1	-	-	-	-
/system.slice/ksmtuned.service	2	-	-	-	-
/system.slice/libstoragemgmt.service	1	-	-	-	-
/system.slice/libvirtd.service	2	-	-	-	-
/system.slice/lvm2-lvmetad.service	1	-	-	-	-
/system.slice/nfs-lock.service	1	-	-	-	-
/system.slice/polkit.service	1	-	-	-	-
/system.slice/postfix.service	3	-	-	-	-
/system.slice/rpcbind.service	1	-	-	-	-
/system.slice/rsyslog.service	1	-	-	-	-
/system.slice/rtkit-daemon.service	1	-	-	-	-
/system.slice/smartd.service	1	-	-	-	-
/system.slice/sshd.service	1	-	-	-	-
/system.slice/systemd-journald.service	1	-	-	-	-
/system.slice/systemd-logind.service	1	-	-	-	-
/system.slice/systemd-udevd.service	1	-	-	-	-
/system.slice/tuned.service	1	-	-	-	-
/system.slice/udisks2.service	1	-	-	-	-
/system.slice/upower.service	1	-	-	-	-

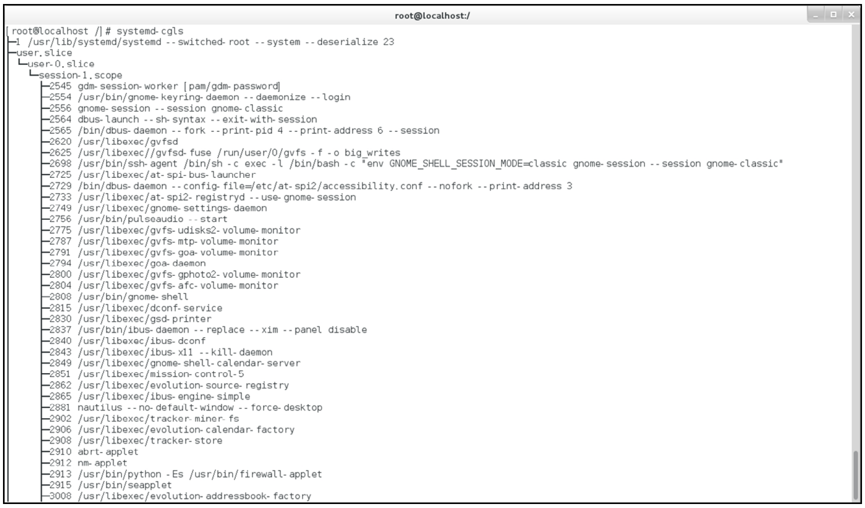
图 8-10　检查资源使用情况

图 8-10 说明这是一个新的类似 top 的程序管理工具，它占会显示每个控制群组中服务的资源使用情况，而非每个程序，这使用户能更轻松地看出哪个服务占用了多少内存、CPU 等。

（8）以树形递归显示选中的 Linux 控制组结构层次。

```
#systemd-cgls
```

显示结果如图 8-11 所示。



```
root@localhost:/
[root@localhost /]# systemd-cgls
├─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 23
├─user.slice
│ └─user-0.slice
│   └─session-1.scope
│     ├─2545 gdm-session-worker [pam/gdm-password]
│     ├─2554 /usr/bin/gnome-keyring-daemon --daemonize --login
│     ├─2556 gnome-session --session gnome-classic
│     ├─2564 dbus-launch --sh-syntax --exit-with-session
│     ├─2565 /bin/dbus-daemon --fork --print-pid 4 --print-address 6 --session
│     ├─2620 /usr/libexec/gvfsd
│     ├─2625 /usr/libexec//gvfsd-fuse /run/user/0/gvfs -f -o big_writes
│     ├─2698 /usr/bin/ssh-agent /bin/sh -c exec -l /bin/bash -c "env GNOME_SHELL_SESSION_MODE=classic gnome-session --session gnome-classic"
│     ├─2725 /usr/libexec/at-spi-bus-launcher
│     ├─2729 /bin/dbus-daemon --config-file=/etc/at-spi2/accessibility.conf --nofork --print-address 3
│     ├─2733 /usr/libexec/at-spi2-registryd --use-gnome-session
│     ├─2749 /usr/libexec/gnome-settings-daemon
│     ├─2756 /usr/bin/pulseaudio --start
│     ├─2775 /usr/libexec/gvfs-udisks2-volume-monitor
│     ├─2787 /usr/libexec/gvfs-mtp-volume-monitor
│     ├─2791 /usr/libexec/gvfs-goa-volume-monitor
│     ├─2794 /usr/libexec/goa-daemon
│     ├─2800 /usr/libexec/gvfs-gphoto2-volume-monitor
│     ├─2804 /usr/libexec/gvfs-afc-volume-monitor
│     ├─2808 /usr/bin/gnome-shell
│     ├─2815 /usr/libexec/dconf-service
│     ├─2830 /usr/libexec/gsd-printer
│     ├─2837 /usr/bin/ibus-daemon --replace --xim --panel disable
│     ├─2840 /usr/libexec/ibus-dconf
│     ├─2843 /usr/libexec/ibus-x11 --kill-daemon
│     ├─2849 /usr/libexec/gnome-shell-calendar-server
│     ├─2851 /usr/libexec/mission-control-5
│     ├─2862 /usr/libexec/evolution-source-registry
│     ├─2865 /usr/libexec/ibus-engine-simple
│     ├─2881 nautilus --no-default-window --force-desktop
│     ├─2902 /usr/libexec/tracker-miner-fs
│     ├─2906 /usr/libexec/evolution-calendar-factory
│     ├─2908 /usr/libexec/tracker-store
│     ├─2910 abrt-applet
│     ├─2912 nm-applet
│     ├─2913 /usr/bin/python -Es /usr/bin/firewall-applet
│     ├─2915 /usr/bin/seapplet
│     ├─3008 /usr/libexec/evolution-addressbook-factory
```

图 8-11　以树形递归显示选中的 Linux 控制组结构层次

（9）显示服务之间的依赖关系，使用命令 systemctl list-deperdencies，如图 8-12 所示。

```
root@localhost:/
[root@localhost /]# systemctl list-dependencies
default.target
├─accounts-daemon.service
├─gdm.service
├─iprdump.service
├─iprinit.service
├─iprupdate.service
├─network.service
├─rtkit-daemon.service
├─systemd-readahead-collect.service
├─systemd-readahead-replay.service
├─systemd-update-utmp-runlevel.service
├─webmin.service
└─multi-user.target
  ├─abrt-ccpp.service
  ├─abrt-oops.service
  ├─abrt-vmcore.service
  ├─abrt-xorg.service
  ├─abrtd.service
  ├─atd.service
  ├─auditd.service
  ├─avahi-daemon.service
  ├─brandbot.path
  ├─chronyd.service
  ├─crond.service
  ├─cups.path
  ├─dbus.service
  ├─hypervkvpd.service
  ├─hypervvssd.service
  ├─iprdump.service
  ├─iprinit.service
  ├─iprupdate.service
  ├─irqbalance.service
  ├─kdump.service
  ├─ksm.service
  ├─ksmtuned.service
  ├─libstoragemgmt.service
  ├─libvirtd.service
  ├─mdmonitor.service
  ├─ModemManager.service
  ├─network.service
  ├─NetworkManager.service
  ├─plymouth-quit-wait.service
```

图 8-12　显示服务之间的依赖关系

另外，还可以使用 systemd 高效地分析和优化引导过程，包括如下几个常用命令。

（1）显示内核和用户空间上一次引导花了多长时间。

```
# systemd-analyze
Startup finished in 3.652s (kernel) + 14.858s (initrd) + 1min 30.079s
(userspace) = 1min 48.590s
```

（2）显示每个服务启动用了多长时间方面的详细信息。

```
# systemd-analyze blame
```

（3）如需知道关于启动过程的更为详细的信息，该程序可创建一个用图表展示单位启动的 SVG 文件。

```
#systemd-analyze plot > plot.svg
```

（4）杀死一个服务进程。

```
# systemctl kill httpd
```

（5）输出运行失败的服务。

```
#systemctl –failed
```

（6）使用 mask 不再加载服务单元或者重新加载服务。

– 不再加载防火墙服务。

```
# systemctl mask firewalld.service
ln -s '/dev/null' '/etc/systemd/system/firewalld.service'
```

– 重新加载防火墙服务。

```
# systemctl unmask firewalld.service
rm '/etc/systemd/system/firewalld.service'
```

3. 其他相关命令

1）设置时区

（1）显示时区列表。

```
#timedatectl list-timezones
Asia/Aden
Asia/Almaty
Asia/Amman
Asia/Anadyr
Asia/Aqtau
...
...
...
Pacific/Wallis
```

（2）设置当前时区。

```
#timedatectl set-timezone Asia/Shanghai
```

（3）设置当前的年月日。

```
# timedatectl set-time YYYY-MM-DD
```

例如：

```
# timedatectl set-time 2014:12:03
```

（4）设置当前的时分秒。

```
# timedatectl set-time HH:MM:SS
```

例如：

```
# timedatectl set-time 11:22:33
```

（5）将硬件时钟配置为地方时。

```
# timedatectl set-local-rtc true
```

（6）重新调整为 UTC。

```
# timedatectl set-local-rtc false
```

2）修改主机名称

（1）临时修改（重启后恢复原来主机名称）。

```
#hostname dlp.server.world
```

（2）永久修改。

```
#hostnamectl set-hostname dlp.server.world
```

（3）查看一下修改结果。

```
   Static hostname: dlp.server.world
         Icon name: computer-vm
           Chassis: vm
        Machine ID: 98a49a78fc9ad91f1b99304c75b94c31
           Boot ID: 09b95ce0bc7f4179b1e8a011ed314c6b
    Virtualization: kvm
  Operating System: CentOS Linux 7 (Core)
       CPE OS Name: cpe:/o:centos:centos:7
            Kernel: Linux 3.10.0-123.4.2.el7.x86_64
      Architecture: x86_64
```

3）本地化设置（locale）

在 Linux 上 locale 用来定义用户所使用的语言。因为 locale 还定义了用户使用的字符集，所以，当语言中含有非 ASCIIA 字符时，设定好正确的 locale 就显得非常重要了。

（1）显示当前的设置。

```
# localectl status
 System Locale: LANG=en_US.UTF-8
 VC Keymap: us
```

```
X11 Layout: n/a
```

（2）显示所有设置列表。

```
$ localectl list-locales
en_AG
en_AG.utf8
en_AU
en_AU.iso88591
en_AU.utf8
en_BW
en_BW.iso88591
en_BW.utf8
output truncated
```

系统默认 locale 记录在 /etc/locale.conf 文件中。
通过如下命令配置为中文。

```
# localectl set-locale LANG="zh_CN.utf8"
```

4）设置键盘布局
（1）显示所有键盘布局列表。

```
# localectl list-keymaps
```

（2）设置自己要使用的键盘布局。

```
# localectl set-keymap en_GB
```

（3）设置 X11 的键盘布局。

```
# localectl set-x11-keymap en_GB
```

8.105 GRUB2

现在 GRand Unified Bootloader（GRUB）已经取代 LILO 在引导加载程序方面的统治地位。GNU GRUB 基于原来的 GRUB 程序（最初由 Erich Stefan Boleyn 所创建），正在由自由软件基金会（Free Software Foundation）进行积极开发。GRUB 不仅可以对各种发行版本的 Linux 进行引导，也能够正常引导 PC 上其他常见的操作系统。由于 GRUB 的功能强大，它已经逐渐取代 Linux 中过去使用的 LILO，而成为各 Linux 发行版本默认的启动引导器。启动引导器是计算机启动过程中运行的第一个真正的软件，通常计算机启动时在通过 BIOS 自检后读取并运行硬盘主引导扇（MBR）中的启动引导器（Boot Loader）程序，启动引导器再负责加载启动硬盘分区中的操作系统。如果启动引导器不能正常工作，将导致操作系统不能正常启动，从而造成计

算机的整体瘫痪，由此可以看出启动引导器在整个计算机启动过程中的重要性。通常每个操作系统在安装过程中都要将自带的启动引导器写入硬盘，以便能够进行自身的引导。在 Red Hat Enterprise Linux 7 中使用 GRUB2 作为默认的启动引导器。GRUB2 虽在名称上是 GRUB 的升级版，但其代码实际上被完全重写了。对于最终用户来讲，GRUB2 带来了若干改进（如表 8-9 所示），如下：

- 自动搜索可用的内核和硬盘中的可用系统。
- 可支持 NTFS 等更多分区格式。
- 可启动 Windows、Mac OS X 等多种系统。
- 更人性化的设计，如输入命令时 Tab 补全很强大、输出很长时会分页显示等。
- 界面的定制更加灵活。

从配置的角度来看，GRUB2 主要有以下几点不同：

- /boot/grub/menu.lst 被/boot/grub/grub.cfg 所取代。
- 主配置文件 grub.cfg 不能直接被用户修改，而是由系统自动生成。
- 每当安装内核、更新 GRUB2 或用户主动运行 grub-mkconfig-o/boot/grub/grub.cfg（曾经是 update-grub）后，GRUB2 会根据用户配置文件重新生成 grub.cfg。
- 可编辑的 GRUB2 配置文件主要包括/etc/default/grub 和/etc/grub.d/下的各文件。
- 分区命名方式有所改变，第一个分区现在是 1 而不是 0，但第一个设备仍然以 0 开始计数，如(hd0,1)表示第一块硬盘上的第一个分区。

表 8-9　GRUB 和 GRUB2 对比

	GRUB（RHEL 6）	GRUB 2（RHEL 7）
配置文件	/boot/grub/grub.conf /boot/efi/EFI/redhat/grub.conf	/boot/grub2/grub.cfg /boot/efi/EFI/redhat/grub.cfg /etc/default/grub /etc/grub.d/*
GRUB 模块文件	/boot/grub/*_stage1_5	/boot/grub2/i386-pc/*.mod /boot/grub2/x86_64-efi/*.mod
安装命令 GRUB	grub-install	grub2-install
安装引导加载程序	无	grub2-mkconfig
加密方法	grub-crypt	grub2-mkpasswd-pbkdf2
设置默认启动	无	grub2-set-defaul

GRUB2 的一些常用命令

1）help

查看命令用法，如 help search 用于查看 search 命令的详细用法。

2）set

设置变量值，如 set default=0、set timeout=5、set root=(hd0,3)等。

需要调用变量 AA 的值时，使用${AA}。如 set root=(hd1,1)，则${root}=(hd1,1)。

3）default

定义默认引导的操作系统。0 表示第一个操作系统，1 表示第二个操作系统，依此类推。

4）timeout

定义在其一时间内用户没有按下键盘上的某个按键，将自动引导 default 指定的操作系统。

5）root

指定用于启动系统的分区。

6）insmod 和 rmmod

加载或移除某模块，如 insmod jpeg、insmod png、insmod ntfs、rmmod png 等。

7）drivemap

drivemap 兼容 GRUB 的 map，主要用于只能从第一硬盘（hd0）引导启动的系统，如 Windows 2000/XP/2003。例如要添加第二硬盘第一分区上的 Windows XP 系统，可使用如下命令：

```
menuentry "Windows XP" {
set root=(hd1,1)
drivemap -s (hd0) ${root}
chainloader +1
}
```

8）ls

列出当前的所有设备，如(hd0)、(hd0,1)、(hd0,5)、(hd1)、(hd1,1)、(hd1,2)等。

```
ls -l
```

详细列出当前的所有设备。对于分区，会显示其 label 及 uuid。

```
ls /
```

列出当前设为 root 的分区下的文件。

```
ls (hd1,1)/
```

列出(hd1,1)分区根目录的文件。

9）search

```
search -f /ntldr
```

列出根目录里包含 ntldr 文件的分区，返回值为分区号。

```
search -l LINUX
```

搜索 LABEL 是 LINUX 的分区。

```
search --set -f /ntldr
```

搜索根目录包含 ntldr 文件的分区并设为 root。注意，如果多个分区含有 ntldr 文件，set 将失去作用。

10）loopback

loopback 命令可用于建立回放设备，例如：

```
loopback lo0 (hd1,1)/abc.iso
```

可以使用 lo0 设备来访问 abc.iso 里的内容，例如，可以从 abc.iso 里的软盘映象中启动。

```
loopback lo0 (hd1,1)/aa.iso
linux (lo0)/memdisk
initrd (lo0)/abc.img
```

要删除某一回放设备，可以使用-d 参数。

```
loopback -d lo0
```

11）pager

分页显示。

```
set pager=1
```

显示满一页时暂停，按 Space 键继续。

```
set pager=0
```

取消分页显示。

12）linux

用 linux 命令取代 GRUB 中的 kernel 命令。

13）chainloader

调用另一个启动器，例如：

```
chainloader (hd0,1)+1
```

调用第一硬盘第一分区引导扇区内的启动器，可以是 Windows 或 Linux 的启动器。

09 Linux 服务器管理命令

9.1 Apache 服务器应用命令

9.1.1 Apache 服务器管理命令

1．apachectl：Apache httpd 服务器控制接口

作用：apachectl 是 Apache httpd 服务器的前端程序，其设计意图是帮助管理员控制 Apache httpd 后台守护进程的功能。

用法：在 sysv 初始化模式中，apachectl 只接受简单的一个单词的命令，如 apachectl command。

主要选项如下。

- start：启动 Apache httpd 后台守护进程。如果已经启动，则产生一个错误。它等价于 apachectl -k start。
- stop：停止 Apache httpd 后台守护进程。它等价于 apachectl -k stop。
- restart：重新启动 Apache httpd 后台守护进程。如果守护进程尚未运行，则启动它。在重新启动守护进程之前，此命令会使用 configtest 自动检查配置文件，以确保 Apache 不会死掉。它等价于 apachectl -k restart。
- fullstatus：显示由 mod_status 提供的完整的状态报告。要使用这个功能，需要启用服务器上的 mod_status 模块，并且系统中有一个基于文本的浏览器，如 Lynx。修改脚本中的 statusurl 变量，可以修改访问状态报告的 URL。
- status：显示一个简要的状态报告。它类似于 fullstatus 选项，但是省略了正在处理的请求的列表。
- graceful：优雅地重新启动 Apache httpd 后台守护进程。如果守护进程尚未启动，则启动它。它和标准重新启动的不同之处在于：不会中断当前已经打开的连接，也不会立即关闭日志。这意味着，如果在日志滚动脚本中使用它，则在处理日志之前必须加入一个实实在在的延迟，以确保旧的日志文件在被处理前已经关闭。在重新启动守护进程之前，此命令会使用 configtest 自动检查配置文件，以确保 Apache 不会死掉。它等价于 apachectl -k graceful。
- graceful-stop：优雅地停止 Apache httpd 后台守护进程。它和标准停止的不同之处在于不

会中断当前已经打开的连接，也不会立即关闭日志。它等价于 apachectl -k graceful-stop。

– configtest：执行一次配置文件语法检查。它解析配置文件，并报告 syntax ok 或者特定的语法错误详细信息。它等价于 apachectl -t。

下列选项仅在 1.3 版本中使用，现在已经被废弃了。

startssl：以支持 ssl 的方式启动 httpd，应当编辑配置文件，并在其中包含与 ssl 支持相关的指令，然后使用 apachectl start 启动服务器。

说明

apachectl 脚本有两种操作模式。首先，作为简单的 httpd 的前端程序，设置所有必要的环境变量，然后启动 httpd，并传递所有的命令行参数。其次，作为 sysv 初始化脚本，接受简单的一个单词的参数，如 start、restart、stop，并把它们翻译为适当的信号发送给 httpd。如果用户的 Apache 安装在非标准的路径中，需要修改 apachectl 脚本，使其中的路径正确地指向 httpd 程序。此外，还可以指定任何必要的 httpd 命令行参数。apachectl 脚本如果执行成功，则返回 0；如果出错，则其返回值大于 0。更多细节可以参见脚本中的注解。

应用实例如下。

（1）启动 Apache 服务器。

```
# apachectl start
httpd (pid 1776) already running
```

（2）重新启动 Apache 服务器。

```
# apachectl restart
```

（3）关闭 Apache 服务器。

```
# apachectl stop
```

（4）检查 Apache 服务器配置文件。

```
# apachectl configtest
syntax ok
```

2．ab：Apache httpd 服务器性能测试工具

作用：ab 是 Apache 超文本传输协议（HTTPD）的性能测试工具，其设计意图是描绘当前所安装的 Apache 的执行性能，主要显示安装的 Apache 每秒可以处理多少个请求。

用法：ab [-a auth-username:password] [-c concurrency] [-c cookie-name=value] [-d] [-e csv-file] [-g gnuplot-file] [-h] [-h custom-header] [-i] [-k] [-n requests] [-p post-file] [-p proxy-auth-username:password] [-q] [-s] [-s] [-t timelimit] [-t content-type] [-v verbosity] [-v] [-w] [-x <table>-attributes] [-x proxy[:port]] [-y <tr>-attributes] [-z <td>-attributes] [http://]hostname[:port]/path

主要选项如下。

- -a auth-username:password：向服务器提供基本认证信息。用户名和密码之间由一个“:”隔开，并将以 Base64 编码形式发送。无论服务器是否需要（即是否发送了 401 认证需求代码），此字符串都会被发送。
- -c concurrency：一次产生的请求个数。默认是一次一个。
- -c cookie-name=value：对请求附加一个“cookie:”头行。其典型形式是 name=value 的一个参数对。此参数可以重复。
- -d：不显示“percentage served within xx [ms] table”消息（为以前的版本提供支持）。
- -e csv-file：产生一个逗号分隔（.csv）文件，其中包含了处理每个相应百分比请求（1%～100%）所需要的相应百分比时间（以微秒为单位）。由于这种用法已经“二进制化”，所以比“gnuplot”用法更有用。
- -g gnuplot-file：把所有测试结果写入一个“gnuplot”或者.tsv（以 Tab 分隔）文件。此文件可以方便地导入到 Gnuplot、IDL、Mathematica、Excel 中。其中的第一行为标题。
- -h：显示使用方法的帮助信息。
- -h custom-header：对请求附加额外的头信息。此参数的典型形式是一个有效的头信息行，其中包含了以冒号分隔的字段和值（如“accept-encoding: zip/zop;8bit”）。
- -i：执行 head 请求，而不是 get。
- -k：启用 keepalive 功能，即在一个 httpd 会话中执行多个请求。默认不启用 keepalive 功能。
- -n requests：在测试会话中所执行的请求个数。默认仅执行一个请求，此时其结果不具有意义。
- -p post-file：包含了 post 数据的文件。
- -p proxy-auth-username:password：对一个中转代理提供基本认证信息。用户名和密码由一个冒号“:”隔开，并将以 Base64 编码形式发送。无论服务器是否需要（即是否发送了 407 代理认证需求代码），此字符串都会被发送。
- -q：如果处理的请求数大于 150，ab 每处理大约 10%或者 100 个请求时，会在 stderr 输出一个进度计数，此-q 标记可以屏蔽这些信息。
- -s：用于编译中（ab -h 会告诉用户）使用了 SSL 的受保护的 HTTPS，而不是 HTTP 协议的时候。此功能是实验性的，最好不要用。
- -s：不显示中值和标准偏差值，而且在均值和中值为标准偏差值的 1～2 倍时，也不显示警告或出错信息。默认会显示最小值、均值、最大值等数值（为以前的版本提供支持）。
- -t timelimit：测试所进行的最大秒数。内部隐含值是“-n 50000”。它可以使对服务器的测试限制在一个固定的总时间以内。默认没有时间限制。
- -t content-type：post 数据时所使用的“content-type”头信息。
- -v verbosity：设置显示信息的详细程度，4 或更大值会显示头信息，3 或更大值可以显示响应代码（如 404、200 等），2 或更大值可以显示警告和其他信息。
- -v：显示版本号并退出。
- -w：以 HTML 表格形式输出结果。默认是白色背景的两列宽度的一张表。
- -x <table>-attributes：设置<table>属性的字符串。

– -x proxy[:port]：对请求使用代理服务器。
– -y <tr>-attributes：设置<tr>属性的字符串。
– -z <td>-attributes：设置<td>属性的字符串。

应用实例如下。

（1）测试本地 Web 服务器。

ab 参数非常多，一般用-c 和-n 参数就行了。例如：

```
ab -c 1000 -n 1000 http://127.0.0.1/index.php
```

这个表示同时处理 1000 个请求并运行 1000 次 index.php 文件。

```
# ab -c 1000 -n 1000 http://127.0.0.1/index.php
this is apachebench, version 2.0.40-dev <$revision: 1.146 $> apache-2.0
copyright 1996 adam twiss, zeus technology ltd, http://www.zeustech.net/
copyright 2006 the apache software foundation, http://www.apache.org/
benchmarking 127.0.0.1 (be patient）
completed 100 requests
completed 200 requests
completed 300 requests
completed 400 requests
completed 500 requests
completed 600 requests
completed 700 requests
completed 800 requests
completed 900 requests
finished 1000 requests
server software:    apache/2.2.3
apache 版本 2.2.3
server hostname: 127.0.0.1
#服务器主机名
server port: 80
#服务器端口
document path: /index.html.zh-cn.gb2312
#测试的页面文件
document length: 1018 bytes
#文件大小
concurrency level: 1000
#并发数
time taken for tests: 8.188731 seconds
#整个测试持续的时间
```

```
complete requests: 1000
#完成的请求数量
failed requests: 0
#失败的请求数量
write errors: 0
total transferred: 1361581 bytes
#整个场景中的网络传输量
html transferred: 1055666 bytes
#整个场景中的 HTML 内容传输量
requests per second: 122.12 [#/sec] (mean)
#系统关心的指标之一，相当于 lr 中的每秒事务数，后面括号中的 mean 表示这是一个平均值
time per request: 8188.731 [ms] (mean)
#系统关心的指标之二，相当于 lr 中的平均事务响应时间，后面括号中的 mean 表示这是一个平均值
time per request: 8.189 [ms] (mean, across all concurrent requests)
#每个请求实际运行时间的平均值
transfer rate: 162.30 [kbytes/sec] received
#平均每秒网络上的流量，能帮助排除是否存在网络流量过大导致响应时间延长的问题
connection times (ms)
min mean[+/-sd] median max
connect: 4 646 1078.7 89 3291
processing: 165 992 493.1 938 4712
waiting: 118 934 480.6 882 4554
total: 813 1638 1338.9 1093 7785
#网络上消耗的时间的分解
percentage of the requests served within a certain time (ms)
50% 1093
66% 1247
75% 1373
80% 1493
90% 4061
95% 4398
98% 5608
99% 7368
100% 7789 (longest request)
```

#整个场景中所有请求的响应情况。在场景中每个请求都有一个响应时间，其中 50%的用户响应时间小于 1093 毫秒，60%的用户响应时间小于 1247 毫秒，最大的响应时间小于 7789 毫秒。由于对于并发请求，CPU 实际上并不是同时处理的，而是按照每个请求获得的时间片逐个轮转处理的，所以基本上第一个 time per request 时间约等于第二个 time per request 时间乘以并发请求数。

（2）以 HTML 表格形式输出结果。

```
ab -w http://localhost/ > 11.html
```

3．httpd：Apache 超文本传输协议服务器

作用：httpd 是 Apache 超文本传输协议（HTTP）服务器的主程序，被设计为一个独立运行的后台进程，它会建立一个处理请求的子进程或线程的池。

用法：httpd [-d serverroot] [-f config] [-c directive] [-c directive] [-d parameter] [-e level] [-e file] [-k start|restart|graceful|stop|graceful-stop] [-r directory] [-h] [-l] [-l] [-s] [-t] [-v] [-v] [-x] [-m]

主要选项如下。

- -d serverroot：将 serverroot 指令设置初始值为 serverroot。它可以被配置文件中的 serverroot 指令所覆盖。其默认值是/usr/local/apache2。
- -f config：在启动中使用 config 作为配置文件。如果 config 不以“/”开头，则它是相对于 serverroot 的路径。其默认值是 conf/httpd.conf。
- -k start|restart|graceful|stop|graceful-stop：发送信号使 httpd 启动、重新启动或停止。
- -c directive：在读取配置文件之前，先处理 directive 的配置指令。
- -c directive：在读取配置文件之后，再处理 directive 的配置指令。
- -d parameter：设置参数 parameter，它配合配置文件中的<ifdefine>段，用于在服务器启动和重新启动时，有条件地跳过或处理某些命令。
- -e level：在服务器启动时，设置 loglevel 为 level。它用于在启动时，临时增加出错信息的详细程度，以帮助排错。
- -e file：将服务器启动过程中的出错信息发送到文件 file。
- -r directory：当在服务器编译中使用了 shared_core 规则时，它指定共享目标文件的目录为 directory。
- -h：输出一个可用的命令行选项的简要说明。
- -l：输出一个静态编译在服务器中的模块的列表。它不会列出使用 loadmodule 指令动态加载的模块。
- -l：输出一个指令的列表，并包含各指令的有效参数和使用区域。
- -m：输出一个已经启用的模块列表，包括静态编译在服务器中的模块和作为 dso 动态加载的模块。
- -s：显示从配置文件中读取并解析的设置结果（目前仅显示虚拟主机的设置）。
- -t：仅对配置文件执行语法检查。程序在语法解析检查结束后立即退出，返回“0”（ok），或者返回非 0 的值（error）。如果还指定了“-d dump_vhosts”，则会显示虚拟主机配置的详细信息。
- -v：显示 httpd 的版本，然后退出。
- -v：显示 httpd 的版本和编译参数，然后退出。
- -x：以调试模式运行 httpd。仅启动一个工作进程，并且服务器不与控制台脱离。

应用实例如下。

（1）显示静态编译在服务器中的模块的列表。

```
# httpd -l
compiled in modules:
core.c
prefork.c
http_core.c
mod_so.c
```

（2）启动 Apache 服务器。

```
# httpd
```

（3）检查 Apache 服务器配置文件。

```
# httpd -t
syntax ok
```

（4）显示配置文件最后一行。

```
# httpd -l > h.log |tail -n 1 h.log
shared object file or library to load into the server at runtime
allowed in *.conf only outside <directory>, <files> or <location>
```

4．htpasswd：管理用于基本认证的用户文件

作用：htpasswd 建立和更新用于基本认证的存储用户名/密码的文本文件。

用法：htpasswd [-c-m] [-d] passwdfile username

　　htpasswd -b [-c] [-m | -d | -p | -s] [-d] passwdfile username password

　　htpasswd -n [-m | -d | -s | -p] username

　　htpasswd -nb [-m | -d | -s | -p] username password

主要选项如下。

- -b：使用批处理方式，也就是直接从命令行获取密码而不进行提醒。使用这个选项需要特别注意，因为命令行中的密码是清晰可见的。
- -c：创建 passwdfile 文件。如果 passwdfile 文件已经存在，那么将被清空并改写。该选项不能和-n 同时使用。
- -n：在标准输出上显示结果而不是更新文件。用于生成可以为 Apache 非文本输出存储用法所接受的密码记录。这个选项改变了命令行语法，因为 passwdfile 参数（通常是第一个）被忽略了。该选项不能和-c 同时使用。
- -m：使用 MD5 加密密码。在 Windows、NetWare、TPF 上这是默认方法。
- -d：使用 crypt()对密码进行加密。在 Windows、NetWare、TPF 以外的平台上，这是默认方法。虽然有可能在所有的平台上被 htpasswd 支持，但是在 Windows、NetWare、TPF 上，该方法不能被 httpd 所支持。
- -s：使用 sha 对密码进行加密。这种方法易于通过 ldap 目录交换用法和 netscape server 进行交换。
- -p：使用明文密码（不加密）。

- -d：如果 username 存在于 passwdfile 文件中，则删除该用户。
- passwdfile：包含用户名和密码的文本文件的名称。如果使用了-c 选项，若文件已存在则更新它，若不存在则创建它。
- username：在 passwdfile 文件中添加或更新记录。若 username 不存在则添加一条记录，若存在则更新其密码。
- password：被加密并存储到文件中的明文密码。必须和-b 选项同时使用。

说明

htpasswd 建立和更新用于基本认证的存储用户名/密码的文本文件。如果 htpasswd 不能读/写此文件，它返回一个出错代码，而不做任何修改。服务器上的资源可以被限制为仅允许由 htpasswd 建立的文件中的用户所访问。此程序只能管理存储在文本文件中的用户名和密码，但是它可以加密并显示密码信息，从而可以为其他数据存储类型所利用。htpasswd 使用专为 Apache 做了修改的 MD5 算法或系统函数 crypt()加密密码。htpasswd 所管理的文件可以包含两种类型的密码。有些用户的密码是用 MD5 加密的，而同一个文件中的其他用户密码则使用 crypt()加密。

应用实例如下。

（1）添加或修改用户 jsmith 的密码。

```
#htpasswd /usr/local/etc/apache/.htpasswd-users jsmith
```

执行上述命令后将提示输入密码。使用 crypt()进行加密。如果指定的文件不存在，htpasswd 将只返回一个错误代码，而不做其他任何事。

（2）使用 htpasswd 添加 Web 访问控制。

首先建立一个存放用户名和密码的目录，这个目录 Web 用户是无法访问到的。例如：

```
#mkdir /www/apache2/passwd
```

然后使用 htpasswd 来生成密码。

```
# ./htpasswd -c /www/apache2/passwd/passwords timo
```

输入后系统会询问 timo 这个用户的密码，这个密码就是以后 Web 访问的密码。

最后修改 httpd.conf 文件，添加如下代码：

```
<directory /www/data>
authtype basic
authname "restricted files"
authuserfile /www/apache2/passwd/passwords
require usertimo
</directory>
```

（3）创建一个新文件并在其中添加一条用户 jane 的记录。

```
#htpasswd -c /home/doe/public_html/.htpasswd jane
```

执行上述命令后将提示输入密码。如果文件存在但是不能被读取或写入，则不会有任何记录被修改，同时 htpasswd 将会显示一个错误信息并返回一个错误代码。

（4）使用 MD5 算法加密。

```
#htpasswd -mb /usr/web/.htpasswd-all jones pwd4steve
```

上面的命令将来自命令行的密码（pwd4steve）使用 MD5 算法加密，并将其存入指定的文件。

安全提示：Web 密码文件（比如由 htpasswd 管理）不应当存在于网络空间中，即不能被客户端有机会访问。所以反对使用-b 选项，因为密码将以明文的形式出现在命令行中。

5．dbmmanage：管理 DBM 用法的用户认证文件

作用：dbmmanage 建立和更新存储用户名及密码的 DBM 用法的文件，以用于 mod_authn_dbm 对 http 用户进行基本认证。Apache http 服务器上的有效资源可以被限制为仅允许由 dbmmanage 建立的文件中的用户所访问。此程序仅用于用户名存储在一个 DBM 文件中的情况下。如果使用文本数据库，请参见 htpasswd。

用法：dbmmanage [encoding] filename add|adduser|check|delete|update username [encpasswd [group[,group...] [comment]]]

dbmmanage filename view [username]

dbmmanage filename import

主要选项如下。

- filename：DBM 用法文件的文件名。一般不带.db、.pag、.dir 后缀。
- username：操作所针对的用户。username 中不能有冒号（:）。
- encpasswd：这是已经加密的密码，用于 update 和 add 命令。使用一个连字符（-）可以显示输入密码的提示，然后输入。另外，在用于 update 命令时，使用一个句号（.）可以保持原有密码不变。
- group：用户所属的组名，组名中不能有冒号（:）。如果不希望指定该用户所属的组，可以使用一个连字符（-），但是需要填写 comment 项。另外，在用于 update 命令时，使用一个句号（.）可以保持原来所属的组不变。
- comment：这是对该用户的说明，如真实姓名、邮件地址等。服务器本身并不使用此信息。
- -d crypt：加密（在 Windows 和 NetWare 以外平台上的默认值）。
- -m md5：加密（在 Windows 和 NetWare 平台上的默认值）。
- -s sha1：加密。
- -p：纯文本（不推荐使用）。

应用实例如下。

（1）在 filename 中增加一个包含 username 和已加密密码 encpasswd 的项。

```
#dbmmanage passwords.dat add rbowen foKntnEF3kSXA
```

（2）输入密码，然后在 filename 中增加一个 username 项。

```
#dbmmanage passwords.dat adduser krietz
```

（3）输入密码，然后检查 filename 中是否存在 username，并且其密码与输入的相同。

```
#dbmmanage passwords.dat check rbowen
```

（4）在 filename 中删除 username 项。

```
#dbmmanage passwords.dat delete rbowen
```

（5）仅显示 DBM 文件的内容；如果指定了 username，则仅显示该用户的信息。

```
#dbmmanage passwords.dat view
```

说明

现在存在许多不同的 DBM 文件用法，用户的系统中也可能存在不止一种支持库，常见的有 SDBM、NDBM、GDBM，Berkeley DB 2。麻烦的是，所有这些库都使用了不同的文件用法，因而必须确保 filename 所采用的用法能够为 dbmmanage 所接受。目前，dbmmanage 无法自己确定所查找的文件的 DBM 类型。如果使用了错误的用法，则简单返回 nothing，或者建立一个不同名称的不同的 DBM 文件，而最坏的情况是，在试图写入这个文件时，可能会破坏该 DBM 文件。dbmmanage 有一个 DBM 用法参数选择列表，在程序前部由@anydbm::isa 数组定义。由于我们更喜欢 Berkeley DB 2 用法，dbmmanage 查找系统库的顺序是 Berkeley DB 2、NDBM、GDBM、SDBM。dbmmanage 会使用第一个找到的库来处理所有的 DBM 文件操作。此顺序与 Perl 中标准的@anydbm::isa 的顺序略微不同。所以，如果要使用任何其他工具来管理 DBM 文件，则必须确保该工具是按此顺序处理的。在用其他语言（如 C）的程序来处理这些文件时，也要考虑这一点。

6．htdigest：管理用于摘要认证的用户文件

作用：htdigest 建立和更新用于摘要认证的、存储用户名/域/密码的文本文件。服务器上的资源可以被限制为仅允许由 htdigest 建立的文件中的用户访问。

用法：htdigest [-c] passwdfile realm username

主要选项如下。

- -c：建立 passwdfile。如果 passwdfile 已经存在，则会首先把它删除。
- passwdfile：包含用户名/域/密码的文件名。若指定了-c 选项，那么，如果该文件不存在则新建，否则先删除然后再新建。
- realm：该用户名所属的域。
- username：在 passwdfile 中所建立或更新的用户名。如果 username 不存在则增加一项，否则改变其密码。

应用实例如下。

建立和更新用于摘要认证的、存储用户名/域/密码的文本文件。

```
# htdigest -cpasswordfile realm username
adding password for username in realm realm.
new password:
re-type new password:
```

7. htcacheclean：清理磁盘缓冲区

作用：htcacheclean 可以用于将 mod_disk_cache 的磁盘缓冲区占用的空间保持在一个合理的水平。这个工具可以手动运行，也可以作为后台守护进程运行。当作为守护进程运行时，它将每隔一段时间检查一次缓冲区所在的目录并进行清理。可以通过 term 或 int 信号停止守护进程的清理操作。

用法：htcacheclean [-d] [-v] [-t] [-r] [-n] -ppath -llimit

htcacheclean -b [-n] [-t] [-i] -dinterval -ppath -llimit

主要选项如下。

- -dinterval：每隔 interval 分钟进行一次清理。这个选项和-D、-v、-r 互斥，不能同时使用。要关闭清理进程，可以使用 sigterm 或 sigint 信号。
- -d：进行一次“演习”而不真正清理任何内容。这个选项和-d 互斥，不能同时使用。
- -v：显示详细的统计信息。这个选项和-d 互斥，不能同时使用。
- -r：进行彻底的清理。它假定 Apache Web 服务器已经停止（否则将在缓冲区中留下垃圾）。这个选项和-d 互斥，不能同时使用。同时该选项隐含了-t 选项。
- -n：温和、精细地进行清理。这样清理过程将会被减慢以有利于其他进程的执行。htcacheclean 有时将会进入休眠状态，以便磁盘 I/O 被延时，或者操作系统内核可以同时执行其他进程。
- -t：删除所有空目录，而默认只删除缓存文件。因为在某些配置情况下会建立数量巨大的目录，这样很可能导致 inode 或文件分配表耗尽。建议使用这个选项。
- -ppath：将 path 指定为磁盘缓冲区的根目录。它必须和 cacheroot 指定的目录相同。
- -llimit：将 limit 指定为磁盘缓冲区允许占用的最大空间。用 xxb 表示 xx 字节，用 xxk 表示 xx 千字节，用 xxm 表示 xx 兆字节。
- -i：智能运行，也就是仅在磁盘缓冲区的内容被更改的情况下运行。仅能够和-d 选项同时使用。

退出状态：htcacheclean 仅在一切操作都成功的情况下返回 0，否则返回 1。

应用实例如下。

每隔 3 分钟进行一次清理。

```
# htcacheclean -dinterval 3
```

8. suexec：在执行外部程序之前切换用户

作用：suexec 使 Apache httpd 服务器在执行 cgi 程序之前切换为另一个用户。为此，它必

须以 root 身份运行。由于 Apache 监听子进程一般不是以 root 身份运行的，所以，可执行文件 suexec 的所有者必须是 root，而且其 setuid 位必须被设置。除了 root，这个文件绝不应该对其他人开放写权限。

用法：suexec -v

主要选项如下。

-v：如果是 root 用户，此选项可以显示 suexec 的编译选项。由于安全方面的原因，所有的配置选项只能在编译时确定。

应用实例如下。

显示 suexec 的编译选项。

```
# suexec -v
 -d ap_doc_root="/var/www"
 -d ap_gid_min=100
 -d ap_httpd_user="apache"
 -d ap_log_exec="/var/log/httpd/suexec.log"
 -d ap_safe_path="/usr/local/bin:/usr/bin:/bin"
 -d ap_uid_min=500
 -d ap_userdir_suffix="public_html"
```

9.1.2　Apache 日志管理命令

1．logresolve：解析 Apache 日志中的 IP 地址为主机名

作用：logresolve 是一个解析 Apache 访问日志中 IP 地址的后处理程序。为了使对名称服务器的影响降到最低，它拥有极为自主的内部散列表缓存，使每个 IP 值仅在第一次从日志文件中读出时才被解析一次。此程序从标准输入设备上获得需要解析的 Apache 日志文件，其中的 IP 地址必须在每行的开始处，行中其余信息必须以空格分隔。

用法：logresolve [-s filename] [-c] < access_log > access_log.new

主要选项如下。

- -s filename：指定记录统计信息的文件名。
- -c：使 logresolve 执行 DNS 验证。在把 IP 地址解析为主机名后，按主机名查找 IP 地址，以验证原地址能否与其中之一相匹配。

应用实例如下。

将 httpd-access.log 解析后，产生 httpd-access.log-resolve 文件。

```
#logresolve-c < httpd-access.log > httpd-access.log-resolve
```

2．rotatelogs：滚动 Apache 日志的管道日志程序

作用：rotatelogs 是一个配合 Apache 管道日志功能使用的简单程序。

用法：rotatelogs [-l] logfile [rotationtime [offset]] | [filesizem]

主要选项如下。

– -l：使用本地时间代替 GMT（格林尼治）时间作为时间基准。注意，在一个改变 GMT（格林尼治）偏移量（比如夏令时）的环境中使用-l 会导致不可预料的结果。
– logfile：日志文件名。有两种用法表示新的日志开始使用的时间。如果 logfile 中包含“%”，则它会被视为用于 strftime()的用法字符串；否则它会被自动加上以秒为单位的“.nnnnnnnnnn”后缀，这里 nnnnnnnnnn 是开始记录日志的格林尼治时间距离 1970 年 1 月 1 日的秒数。常用的字符串如下。
  - %y：4 位数的年份。
  - %m：2 位数的月份。
  - %d：2 位数的一个月中的日期数。
  - %h：2 位数的小时数（24 小时制）。
  - %m：2 位数的分钟数。
  - %s：2 位数的秒数。
  - %u：2 位数的一年中的星期数（星期天为一周的第一天）。
  - %w：2 位数的一年中的星期数（星期一为一周的第一天）。
  - %w：1 位数的星期几（星期天为一周的第一天）。
– rotationtime：日志文件滚动的以秒为单位的间隔时间。
– offset：相对于 UTC 的时差的分钟数。如果省略，则假定为“0”并使用 UTC 时间。例如，要指定 UTC 时差为“–5 小时”的地区的当地时间，则此参数应为“–300”。
– filesizem：指定以 filesizem 文件大小滚动，而不是按照时间或时差滚动。

应用实例如下。

在日志文件大小增长到 5MB 时滚动该日志。

```
customlog "|bin/rotatelogs /var/logs/logfile 5m" common
```

3．ApacheTop 工具：分析 Web 日志

作用：作为网站管理员，我们经常需要知道当前什么人正在访问我们的网站，谁正在频繁地抓取我们网站的内容，什么搜索引擎正在抓取我们的网站？面对这些问题，我们虽然可以去查看 Log 日志文件，但是却不能让我们实时统计，不能给我们直观的统计数据。有了 ApacheTop 这个工具，我们就可以实时地跟踪 Log 的变化，对网站管理帮助很大。ApacheTop 是一个 Apache 的 Log 实时监测程序，能查看访问者正在查看哪些文件，以及访问者 IP 等信息。

用法：ApacheTop 选项

主要选项如下。

– -f logfile：打开日志文件（默认是/var/log/httpd-access.log 文件）。

url/host/referrer 选项如下。

– -q：保持请求字符串。
– -l：所有的 URL 小写。
– -s num：保持 URL 路径段数量。
– -p：在 referrer 前面保留协议。
– -r：保留每一个的主机/IP。

默认设置为保持记录状态 30 秒，30 秒后数据刷新。为了提供更多的信息方便用户进行分析，可以调节下面的参数。

– -h hits：保持状态只到多少点击数。
– -t secs：保持状态只到多少秒。
– -d secs：刷新延迟时间。
– -h：显示帮助。

应用实例如下。

如果希望查看访问日志，可以使用如下命令：

```
#apachetop -f /var/log/access_log
```

如图 9-1 所示是 ApacheTop 初始工作界面。

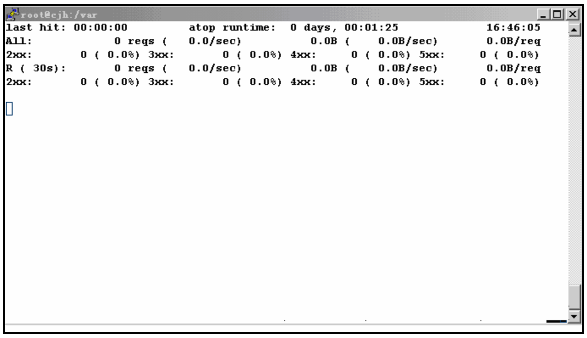


图 9-1　ApacheTop 初始工作界面

ApacheTop 和 Linux 的 top 命令类似，使用交互界面，按 h 键显示帮助界面，如图 9-2 所示。

– 在 ApacheTop 中使用 d 键在 urls、referrers、hosts 间切换。
– 在 ApacheTop 中使用 n 键在 hits 和 bytes 间切换。
– 在 ApacheTop 中使用 q 键退出。
– 在 ApacheTop 中使用 p 键暂停显示。

另外，ApacheTop 还有 3 个子菜单。

（1）按小写 s 键进入排序界面，如图 9-3 所示，排序顺序键：

r) REQUESTS　R) REQS/SEC　b) BYTESB　) BYTES/SEC　2) 2xx　3) 3xx　4) 4xx　5) 5xx

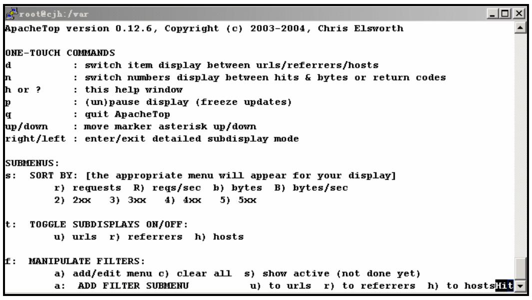


图 9-2　ApacheTop 的帮助界面

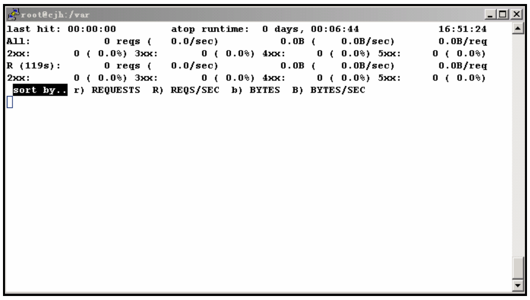


图 9-3　排序界面

（2）按小写 t 键进入切换界面，如图 9-4 所示，切换键：

u) URLS　r) REFERRERS　h) HOSTS

（3）按小写 f 键进入操作过滤界面，如图 9-5 所示，操作过滤键：

a）编辑菜单　c）清除所有信息　s）显示

a：增加过滤子菜单

u) to urls　r) to referrers　h) to hosts

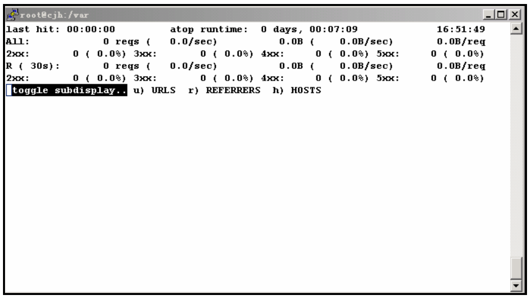


图 9-4　切换界面

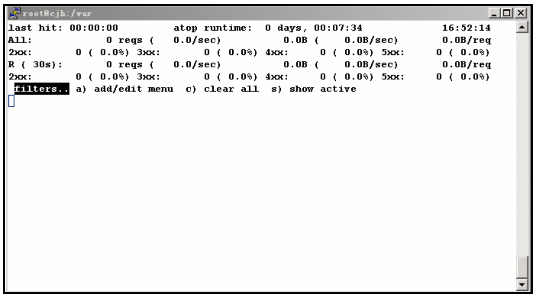


图 9-5　过滤界面

如图 9-6 所示是 ApacheTop 工作 2 小时后的一个工作截图。这里使用 PuTTY 远程登录到 Linux 服务器运行的 ApacheTop。

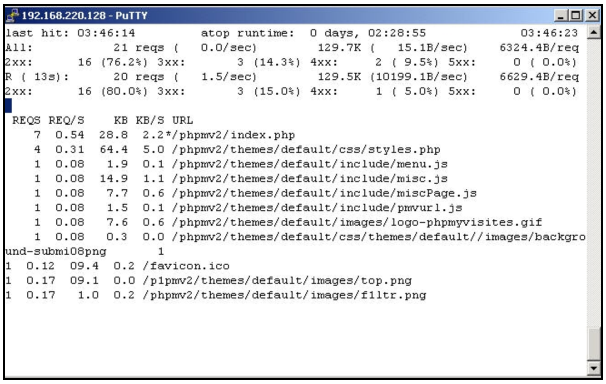


图 9-6　ApacheTop 工作 2 小时后的一个工作截图

4．使用 webalizer 工具

webalizer 是一个高效的、免费的 Web 服务器日志分析程序。其分析结果以 HTML 文件用法保存，从而可以很方便地通过 Web 服务器进行浏览。Internet 上的很多站点都使用 webalizer 进行 Web 服务器日志分析。webalizer 具有以下一些特性。

- 它是用 C 语言编写的程序，所以其具有很高的运行效率。在主频为 200MHz 的机器上，webalizer 每秒可以分析 10 000 条记录，所以分析一个 40MB 的日志文件只需要 15 秒。
- webalizer 支持标准的一般日志文件用法（Common Logfile Format）。除此之外，也支持几种组合日志用法（Combined Logfile Format）的变种，从而可以统计客户情况及客户

操作系统类型。并且现在 webalizer 已经可以支持 wu-ftpd xferlog 日志用法及 squid 日志文件用法。
- 支持命令行配置及配置文件。
- 可以支持多种语言，也可以自己进行本地化工作。
- 支持多种平台，如 UNIX、Linux、NT、OS/2 和 Mac OS 等。

RHEL 5.0 已经包括 webalizer 工具，也可以通过 http://www.mrunix.net/webalizer/下载安装。

可以通过命令行配置 webalizer，也可以通过配置文件进行配置，本书中将介绍如何使用命令行参数进行配置。

用法：webalizer [选项] [logfile]

主要选项如下。

- -h：打印帮助信息。
- -v -v：打印版本信息。
- -d：打印附加调试信息。
- -f type：日志用法类型。type= (clf | ftp | squid)。
- -i：忽略历史文件。
- -p：保留状态（递增模式）。
- -q：忽略消息信息。
- -q：忽略所有信息。
- -y：忽略国家图形。
- -g：忽略小时统计图形。
- -h：忽略小时统计信息。
- -l：忽略彩色图例。
- -l num：在图形中使用数字背景线。
- -m num：访问超时（seconds）。
- -t：打印时间信息。
- -c file：指定配置文件。
- -n name：使用的主机名。
- -o dir：结果输出目录。
- -t name：指定报告题目上的主机名。
- -a name：隐藏用户代理名称。
- -r name：隐藏访问链接。
- -s name：隐藏客户。
- -u name：隐藏 URL。
- -x name：使用文件扩展名。
- -p name：页面类型扩展名。
- -i name：index 别名。
- -a num：显示前几名客户类型。
- -c num：显示前几名国家。

– -r num：显示前几名链接。
– -s num：显示前几名客户。
– -u num：显示前几名 URLs。
– -e num：显示前几名访问页面。
– -e num：显示前几名不存在的页面。
– -x：隐藏个别用户。
– -d name：使用 DNS 缓存文件。
– -n num：DNS 进程数（0=禁用 DNS）。

应用实例如下。

分析 Web 日志如下。

Web 服务器主机名为 www.test.com，统计站点域名为 www.test.com，访问日志为/var/log/httpd/access_log，将 webalizer 分析结果输出到/var/www/html/log 下面，则可以建立以下脚本/etc/rc.d/webalizer。

```
#!/bin/sh
run=/usr/sbin/webalizer
$run -f clf -p -n ' ' -t 'www.test.com' -o /var/www/html/log /var/log/httpd/access_log
```

说明如下。

– -f clf：指明我们的 Web 日志用法为标准的一般日志文件用法（Common Logfile Format）。
– -p：指定使用递增模式，也就是说每进行一次分析后，webalizer 会生产一个历史文件，下一次分析时就可以不分析已经处理过的部分。这样就可以在短时间内转换日志文件，而不用担心访问量太大时日志文件无限增大。
– -n''：指定服务器主机名为空，这样输出结果会美观一些。
– -t'www.test.com'：指定输出结果标题。
– /var/log/httpd/access_log：指定日志文件。

然后在/etc/crontab 中加入如下代码：

```
01 1 * * * root /etc/rc.d/webalizer
```

即每天凌晨 1 点执行该脚本。

接着运行/etc/rc.d/init.d/crond reload 重载入 crond 服务。

测试如下。

执行以下命令：

```
# /etc/rc.d/webalizer
```

最后在浏览器中访问 http://www.test.com/log/，就可以看到 webalizer 的分析结果了。

9.2 DNS 服务器管理命令

9.2.1　named：域名服务器管理命令

作用：named 是 Internet 域名服务器。named 将读取默认的配置文件/etc/named.conf，读取任何初始数据并监听查询。named 需要超级用户权限才能执行。

用法：named [-4|-6] [-c conffile] [-d debuglevel] [-f|-g] [-n number_of_cpus] [-p port] [-s] [-t chrootdir] [-u username] [-d]

主要选项如下。

- -4：仅使用 IPv4 技术。
- -6：仅使用 IPv6 技术。
- -c：conffile 使用指定的 config_file 而非/etc/named.conf。
- -d：debuglevel 输出调试信息。d 之后的数字确定所输出消息的级别。如果为负数，则 debuglevel 将设置为“1”。
- -f：在前台运行此进程，不要 fork(2)和作为守护程序运行（默认值是作为守护程序运行）。
- -g：指定在前台运行 named 时，将所有的日志记录都记录到 stderr 中。
- -n no_of_cpus：该选项可用来创建等于 no_of_cpus 的工作线程，以便可以利用多个 CPU。如果不指定此选项，则 named 将尝试确定存在的 CPU 数量，并为每个 CPU 创建一个线程。如果 named 无法确定 CPU 的数量，则将创建单个工作线程。
- -p port_number：使用另一个端口号。默认端口号是 53。
- -t chrootdir：指定服务器应当运行的工作目录。配置文件的 options 语句的 chrootdir 子句将覆盖在命令行上指定的任何值。默认工作目录是当前目录（.）。
- -u username：指定服务器在进行初始化之后应当以哪个用户的身份运行。指定的值可以是用户名或数字用户 ID。
- -v：报告版本号并退出。
- -x cache-file：从高速缓存文件中将数据装入默认视图的高速缓存。注意，本选项并非必须使用，它只对 Bind 9 开发者有意义。
- -d：支持动态 DNS。

应用实例如下。

（1）使用 RPM 软件包。

RHEL 5.0 提供了 Bind DNS 服务器的 RPM 软件包，如果使用这种方法，安装 DNS 服务器、配置相关配置文件后，启动、停止、重新启动命令依次是：

```
启动 named
# service named start
启动 named：                        [确定]
关闭 named
# service named stop
```

```
停止 named：                  [确定]
重新启动 named
停止 named：                  [确定]
启动 named：                  [确定]
```

如果希望 DNS 服务器随系统自动启动，可以使用如下命令：

```
#chkconfig named on
```

（2）使用源代码安装 Bind DNS 服务器。

如果使用 http://www.isc.org/products/bind/下载源代码，安装 Bind DNS 服务器、配置相关配置文件后，启动、停止、重新启动命令依次是：

```
# /etc/init.d/named start
# /etc/init.d/named stop
# /etc/init.d/named restart
```

（3）查看 Bind DNS 版本。

```
# named -v
bind 9.3.3rc2
```

9.2.2　rndc：DNS 服务器控制

作用：rndc（Remote Name Daemon Control）用于系统管理员控制 Bind 的运行。它替换了 Bind 8 中的 ndc 命令工具。rndc 可以完成如下任务。

- 重新加载配置文件。
- 查看 named 当前运行状态。
- 转储服务器缓存信息。
- 将服务器转入调试模式。

用法：rndc [-c config-file] [-s server] [-p port] [-k key-file] [-y keyid] [-v] {command}

主要选项如下。

- -c config-file：指定 rndc 的配置文件，默认为/etc/bind/rndc.conf。
- -s server：指定一个服务器来匹配 rndc 配置文件中的 server 声明；若不指定，则匹配 option 选项中的 default-server 子句。
- -p port：指定 rndc 控制通道的 TCP 端口，默认为 953。
- -k key-file：指定 key-file，在 cjn 中默认为/etc/bind/rndc.key。
- -y keyid：指定一个 keyid 来匹配 rndc 配置文件中的 key 声明；若不指定，则匹配 server 声明中的 key 子句；若 server 声明中没有 key 子句，则匹配 option 选项中的 default-key 子句。
- -v：显示更多信息。

command 子命令如下。

Ø reload：重新加载主配置文件和所有区文件。

- reconfig：重新加载 namd.conf 配置文件和所有新的区，但不会重新加载已存在的区文件，即使区文件已经被修改了也不会加载，特别是当有大量区的时候。这比使用 reload 要快很多，因为这避免了对区文件修改时间的检查。
- stats：将统计信息写入统计文件 named.stats 中。
- querylog：启动用户请求的日志记录。请求的日志记录也可以通过在 named.conf 中 logging 部分的 queries category 和 channel 来启动。
- dumpdb：把服务器缓存中的信息转储到 named_dump.db 文件中。
- stop：服务器停机，且保存未做完的更新。
- halt：服务器立即停机，不保存未做完的更新。
- trace：将服务器的调试等级加 1。
- trace level：直接设置服务器的调试等级。
- notrace：将服务器的调试等级设为 0，即关闭调试。
- flush：清除域名服务器缓存中的内容。
- status：显示当前运行的 named 的状态。

应用实例如下。

（1）重新加载主配置文件和所有区文件。

```
#rndc reload
```

（2）停止 Bind 的运行。

```
#rndc stop
```

（3）将统计信息写入统计文件 named.stats 中。

```
#rndc stats
```

（4）显示当前运行的 named 的状态。

```
#rndc status
```

（5）把服务器缓存中的信息转储到 named_dump.db 文件中。

```
#rndc dumpdb
```

9.2.3　named-checkconf：检查 DNS 配置

作用：named-checkconf 检查 DNS 配置文件 named.conf 的语法。

用法：named-checkconf[-t directory] [filename]

主要选项如下。

- -t directory：当使用 chroot jail 方式运行 Bind 时，指定 chroot 目录。
- filename：指定主配置文件，默认为/etc/bind/named.conf。

应用实例如下。

检查 DNS 配置文件 named.conf 的语法。

```
# named-checkconf
```

若无任何输出，则表示 named.conf 文件的语法正确。

下面是一个检查出现错误的例子。

```
# named-checkconf /etc/named.conf
/etc/named.conf:23: unknown option 'fi1e'
```

上面的信息说明在第 23 行有一个错误语句，原来是把“file”误拼写为了“fi1e”。找到错误原因后，用 vi 修改配置文件，就可以很快地排除故障。

9.2.4 named-checkzone：检查区域文件的合法性

作用：named-checkzone 通过检查句法的正确性来检查区域（Zone）文件的正确性。named-checkzone 如果没有检查到错误，会简单地返回一个“ok”字符。

用法：named-checkzone [-d] [-q] [-t directory] zonename filename

主要选项如下。

- -d：用于打开调试模式。
- -q：指定为“安静”模式，不显示正确的输出信息。
- -t directory：当使用 chroot jail 方式运行 Bind 时，指定 chroot 目录。
- zone：指定要检查的区。
- filename：指定区文件的路径。

应用实例如下。

要检查 cjh.net 区配置的合法性，可以使用如下命令：

```
named-checkzone cjh.net /etc/bind/ cjh.net.hosts
```

下面是一个检查出现错误的例子。

```
#named-checkzone -q /var/named/localhost.zone
dns_zone_load: couldnot find ns and/or soa records or bad datted quad
```

显示没有定义 soa 记录或 a 记录设定了一个错误的 IP 地址。查看/var/named/localhost.zone 文件后发现如下一行：

```
1d in a127.0.0.l
```

这显然是一个拼写错误，“i”错误地拼写为“l”。找到错误文件并修改后，就可以很快地排除故障。

9.2.5 dig：发送域名查询信息包到域名服务器

作用：dig（Domain Information Groper，域名信息搜索）是一个灵活的命令行工具，它可以用来从域名系统服务器中收集信息。

dig 有两种模式：简单交互模式用于简单的查询；而批处理模式则可以对包含多个查询条

目的列表执行查询。所有查询选项都可以从命令行输入，通常简单的 dig 用法为 dig @server name type。

主要选项如下。

- server：待查询名称服务器的名称或 IP 地址。可以是用点分隔的 IPv4 地址或用冒号分隔的 IPv6 地址。当由主机提供服务器参数时，dig 在查询域名服务器前先解析那个名称。如果没有服务器参数可以提供，dig 参考/etc/resolv.conf，然后查询列举在那里的域名服务器。显示来自域名服务器的应答。
- name：将要查询的资源记录的名称。
- type：显示所需的查询类型——any、a、mx、sig，以及任何有效查询类型等。如果不提供任何类型参数，dig 将对记录 a 执行查询。

dig 提供查询选项号，它影响搜索方式和结果显示。一部分决定请求报头设置或复位标志位，一部分决定显示哪些回复信息，其他的确定超时和重试战略。每个查询选项被带前缀（+）的关键字标识。一些关键字设置或复位一个选项。通常前缀是求反关键字含义的字符串 no。其他关键字分配各选项的值，比如超时时间间隔。它们的用法形如+keyword=value。查询的主要选项如下。

- +[no]tcp：查询域名服务器时使用[不使用]TCP。默认行为是使用 UDP，除非是 axfr 或 ixfr 请求，才使用 TCP 连接。
- +[no]vc：查询名称服务器时使用[不使用]TCP。+[no]tcp 的备用语法提供了向下兼容。vc 代表虚电路。
- +[no]ignore：忽略 UDP 响应的中断，而不是用 TCP 重试。默认情况下运行 TCP 重试。
- +domain=somename：设定包含单个域 somename 的搜索列表，好像被/etc/resolv.conf 中的域伪指令指定，并且启用搜索列表处理，好像给定了+search 选项。
- +[no]search：使用[不使用]搜索列表或 resolv.conf 中的域伪指令（如果有的话）定义的搜索列表。默认情况下不使用搜索列表。
- +[no]aaonly：该选项不做任何事。它用来提供对设置成未实现解析器标志的 dig 的旧版本的兼容性。
- +[no]adflag：在查询中设置[不设置]AD（真实数据）位。目前 AD 位只在响应中有标准含义，而在查询中没有，但是出于完整性考虑，在查询中这种性能可以设置。
- +[no]cdflag：在查询中设置[不设置]CD（检查禁用）位。它请求服务器不运行响应信息的 DNSSEC 合法性。
- +[no]recursive：切换查询中的 RD（要求递归）位设置。在默认情况下设置该位，也就是说 dig 正常情形下发送递归查询。当使用查询选项+nssearch 或+trace 时，递归自动禁用。
- +[no]nssearch：这个选项被设置时，dig 试图寻找包含待搜名称的网段的权威域名服务器，并显示网段中每台域名服务器的 SOA 记录。
- +[no]trace：切换为待查询名称从根名称服务器开始的代理路径跟踪。默认情况下不使用跟踪。一旦启用跟踪，dig 使用迭代查询解析待查询名称。它将按照从根服务器的参照，显示来自每台使用解析查询的服务器的应答。
- +[no]cmd：设定在输出中显示指出 dig 版本及其所用的查询选项的初始注释。默认情况

下显示注释。

- +[no]short：提供简要答复。默认值是以冗长用法显示答复信息。
- +[no]identify：当启用+short 选项时，显示[不显示]提供应答的 IP 地址和端口号。如果请求简短用法应答，默认情况下不显示提供应答的服务器的源地址和端口号。
- +[no]comments：切换输出中的注释行显示。默认值是显示注释。
- +[no]stats：该查询选项设定显示统计信息，包括查询进行时、应答的大小等。默认显示查询统计信息。
- +[no]qr：显示[不显示]发送的查询请求。默认不显示。
- +[no]question：当返回应答时，显示[不显示]查询请求的问题部分。默认作为注释显示问题部分。
- +[no]answer：显示[不显示]应答的回答部分。默认显示。
- +[no]authority：显示[不显示]应答的权限部分。默认显示。
- +[no]additional：显示[不显示]应答的附加部分。默认显示。
- +[no]all：设置或清除所有显示标志。
- +time=t：为查询设置超时时间为 *t* 秒。默认是 5 秒。如果将 *t* 设置为小于 1 的数，则以 1 秒作为查询超时时间。
- +tries=a：设置向服务器发送 UDP 查询请求的重试次数为 *a*，代替默认的 3 次。如果 *a* 小于或等于 0，则采用 1 作为重试次数。
- +ndots=d：出于安全考虑，设置必须出现在名称 d 的点数。默认值是使用在/etc/resolv.conf 中的 ndots 语句定义的，或者是 1（如果没有 ndots 语句的话）。带更少点数的名称被解释为相对名称，并通过搜索列表中的域或文件/etc/resolv.conf 中的域伪指令进行搜索。
- +bufsize=b：设置使用 EDNS0 的 UDP 消息缓冲区大小为 *b* 字节。缓冲区的最大值和最小值分别为 65 535 字节和 0 字节。超出这个范围的值自动舍入到最近的有效值。
- +[no]multiline：以详细的多行用法显示类似 SOA 的记录，并附带可读注释。默认值是每单个行上显示一条记录，以便于计算机解析 dig 的输出。

说明

dig（域信息搜索器）命令是一个用于询问 DNS 域名服务器的灵活的工具。它执行 DNS 搜索，显示从接受请求的域名服务器返回的答复。多数 DNS 管理员利用 dig 对 DNS 进行故障诊断，因为它灵活性好、易用、输出清晰。虽然通常情况下 dig 使用命令行参数，但它也可以按批处理模式从文件读取搜索请求。不同于早期版本，dig 的 Bind 9 实现允许从命令行发出多个查询。除非被告知请求特定域名服务器，dig 将尝试/etc/resolv.conf 中列举的所有服务器。当未指定任何命令行参数或选项时，dig 将对“.”（根）执行 ns 查询。

应用实例如下。

（1）查询 cjn.org.cn 的所有 rr 信息。

```
$ dig cjn.org.cn any

; <<>> dig 9.3.2 <<>> cjn.org.cn any
;; global options:  printcmd
;; got answer:
;; ->>header<<- opcode: query, status: noerror, id: 14997
;; flags: qr rd ra; query: 1, answer: 3, authority: 2, additional: 3

;; question section:
;cjn.org.cn.              in     any

;; answer section:
cjn.org.cn.        10676   in     a      211.148.131.7
cjn.org.cn.        10676   in     ns     dns8.hichina.com.
cjn.org.cn.        10676   in     ns     dns7.hichina.com.

;; authority section:
cjn.org.cn.        10676   in     ns     dns7.hichina.com.
cjn.org.cn.        10676   in     ns     dns8.hichina.com.

;; additional section:
dns7.hichina.com.      172625  in     a      218.30.103.247
dns7.hichina.com.      172625  in     a      218.30.103.250
dns8.hichina.com.      172625  in     a      218.244.143.41

;; query time: 3 msec
;; server: 192.168.0.191#53(192.168.0.191)
;; when: mon jul 10 17:57:45 2006
;; msg size  rcvd: 1722
```

（2）查询 cjn.com 的 mx 记录及其相关信息。

```
$ dig @ns.cjn.com  cjn.com mx

; <<>> dig 9.3.2 <<>> @ns.cjn.com cjn.com mx
; (1 server found)
;; global options:  printcmd
;; got answer:
;; ->>header<<- opcode: query, status: noerror, id: 35114
;; flags: qr aa rd; query: 1, answer: 1, authority: 3, additional: 2
```

```
;; question section:
;cjn.com.                 in     mx

;; answer section:
cjn.com.           3600    in     mx     10 fiordland.cjn.com.

;; authority section:
cjn.com.           10800   in     ns     ns1.blackcatnetworks.co.uk.
cjn.com.           10800   in     ns     ns.cjn.com.
cjn.com.           10800   in     ns     ns0.blackcatnetworks.co.uk.

;; additional section:
fiordland.cjn.com.   1800    in     a      82.211.81.145
ns.cjn.com.           600     in     a      82.211.81.173

;; query time: 355 msec
;; server: 82.211.81.173#53(82.211.81.173)
;; when: mon jul 10 17:58:52 2006
;; msg size  rcvd: 1613
```

9.2.6 nslookup：交互式查询名称服务器

作用：nslookup 命令用于验证与 DNS 名称服务器的通信。

nslookup 有两种运行模式：交互式和非交互式。交互式模式允许向名称服务器查询多台主机的域，或者打印出一个域中的主机列表。非交互式模式仅仅用来显示一个关于主机或域名的名字和被请求的信息。

nslookup 命令的用法如下。

用于交互模式：nslookup [- server]。

用于非交互模式：nslookup <hostname | ipaddr> [server]。

不使用任何参数，将进入交互模式（查询时使用 resolv.conf 中指定的域名服务器）。若指定 server 参数，则使用指定的 DNS 服务器响应查询。若不指定此参数，则使用 resolv.conf 中指定的域名服务器。若查询时想指定 DNS 服务器，则第一个参数必须是“-”，第二个参数是一台 DNS 的主机名或 IP 地址。退出交互模式可以使用 exit 命令。设置 rr 的查询类型可以使用 set type=xxx 命令，其中 xxx 是 rr 的类型，如 soa、a、cname、ns 等；还可以是 any，表示所有记录类型。

应用实例如下。

（1）更改默认查询类型为主机信息（hinfo），并且初始超时时间为 10 秒。

```
#nslookup -query=hinfo -timeout=10
```

（2）设置域和搜索列表为 3 个名称：lcs.mit.edu、ai.mit.edu 和 mit.edu。

```
nslookup -set srchlist=lcs.mit.edu/ai.mit.edu/mit.edu
```

该命令覆盖默认域名和 set domain 命令的搜索列表，使用 set all 命令显示列表。

（3）确定名称是否指定主机、域或其他实体。

```
nslookup -querytype=any austin.ibm.comnslookup
```

该命令返回有关名称 austin.ibm.com 的所有可用的信息，包括权限语句（SOA）、名称服务器、邮件交换器和主机因特网地址信息，如下：

```
server: benames.austin.ibm.com
address: 9.3.199.2

austin.ibm.com origin = ausname1.austin.ibm.com
      mail addr = brian.chriss.austin.ibm.com
      serial=1993081210,refresh=3600,retry=300,expire=604800, min=86400
austin.ibm.comnameserver = ausname1.austin.ibm.com
austin.ibm.comnameserver = bb3names.austin.ibm.com
austin.ibm.comnameserver = benames.austin.ibm.com
austin.ibm.comnameserver = b45names.austin.ibm.com
austin.ibm.comnameserver = bbcnames.austin.ibm.com
austin.ibm.comnameserver = netmail.austin.ibm.com
austin.ibm.compreference = 10, mail exchanger = netmail.austin.ibm.com
austin.ibm.cominet address = 129.35.208.98
ausname1.austin.ibm.cominet address = 129.35.17.2
bb3names.austin.ibm.cominet address = 129.35.208.99
benames.austin.ibm.com inet address = 9.3.199.2
b45names.austin.ibm.cominet address = 129.35.49.2
bbcnames.austin.ibm.cominet address = 129.35.17.68
netmail.austin.ibm.com inet address = 129.35.208.98
```

（4）在 opus 主机上执行非交互式查询。

```
=#nslookup opus
```

nslookup 命令与 host 命令的应答相同。该命令返回域名和 opus 主机的因特网地址，如下：

```
name: opus.austin.ibm.com
address: 129.35.129.223
```

如果主机 opus 已经是名称服务器（以空/etc/resolv.conf 文件正在运行 named 守护程序的主机），将会显示下列信息：

```
server: loopback
address: 0.0.0.0
```

nslookup 命令退出状态如下。

如果查询请求不成功，则 nslookup 命令返回下列错误消息之一。

- timed out：表示服务器在超出指定的重试次数后不响应请求。
- no response from server：表示名称服务器不在服务器上运行。
- no records：表示服务器没有主机的指定查询类型的资源记录，即使主机名有效。
- non-existent domain：表示主机或域名不存在。
- connection refused：表示在查询时，不能连接到名称或远程用户信息服务命令服务器。该错误是与 ls 和 finger 请求有关的典型错误。
- network is unreachable：表示在查询时，不能连接到名称或远程用户信息服务命令服务器。该错误是与 ls 和 finger 请求有关的典型错误。
- server failure：表示名称服务器遇到内部矛盾并且不能返回有效的应答。
- refused：表示名称服务器拒绝请求服务。
- format error：表示名称服务器拒绝数据包请求，因为用法不正确。

9.2.7　host：使用域名服务器查询主机名字

作用：host 是一个相对较新且简单的 DNS 查询工具。默认状态下，host 命令将只对主机名和 IP 地址进行转换，但是此项功能可以通过选项的使用得到拓展。

用法：host [-v] [-t type | -a] <hostname|ipaddr> [server]

主要选项如下。

- -v：用于显示更多的信息。
- -t：用于指定 rr 的类型，-a 表示 any。
- hostname：指定要查询的主机名。
- ipaddr：指定要查询的 IP 地址。
- server：若指定 server 参数，则使用指定的 DNS 服务器响应查询；若不指定此参数，则使用 resolv.conf 中指定的域名服务器响应查询。

应用实例如下。

（1）使用 host 进行正向域名查询。

```
# host www.sina.com.cn
www.sina.com.cn is an alias for jupiter.sina.com.cn.
jupiter.sina.com.cn is an alias for antares.sina.com.cn.
antares.sina.com.cn has address 218.30.66.54
antares.sina.com.cn has address 218.30.66.121
antares.sina.com.cn has address 218.30.66.127
```

```
    www.sina.com.cn is an alias for jupiter.sina.com.cn.
    jupiter.sina.com.cn is an alias for antares.sina.com.cn.
    www.sina.com.cn is an alias for jupiter.sina.com.cn.
    jupiter.sina.com.cn is an alias for antares.sina.com.cn.2
```

（2）使用 host 进行正向 IP 地址查询。

```
    # host 62.140.195.84
    84.195.140.62.in-addr.arpa domain name pointer ns1.mnet.net.uk.3
```

（3）使用 host 查询指定的 rr 记录。

```
    $ host -t soa cjn.com
    cjn.com has soa record ns.cjn.com. hostmaster.canonical.com. 2006070705
10800 3600 604800 3600
    $ host -t mx cjn.com
    cjn.com mail is handled by 10 fiordland.cjn.com.
```

（4）列出 rutgers.edu 域中所有的主机。

```
    #host -l -v -t any rutgers.edu
```

上面的命令以正式主域文件格式下载 rutgers.edu 的全部区域（Zone）数据（然而由于某些复杂原因，SOA 记录显示了两次）。注意：“-l” 执行全部的区域传输，然后过滤出要求的信息，所以万不得已才用这个命令。

9.2.8　dnssec-keygen：DNSSEC 密钥生成工具

作用：dnssec-keygen 命令为 DNSSEC（安全 DNS）生成密钥，也可以生成与 TSIG（事务特征符）一起使用的密钥。

用法：dnssec-keygen [-a algorithm] [-b keysize] [-n nametype] [-c class] [-e] [-g generator] [-h] [-p protocol] [-r randomdev] [-s strength] [-t type] [-v level] [name]

主要选项如下。

- -a algorithm：选择加密算法。algorithm 的值必须为 rsamd5、rsa、dsa、dh（diffie hellman）或 hmac-md5 之一。这些值不区分大小写。注意，对于 DNSSEC 而言，dsa 是一个强制性的实现算法，推荐使用 rsa。对于 TSIG 而言，hmac-md5 是强制性的。
- -b keysize：指定密钥位数。密钥大小的选项取决于所用的算法。rsa 密钥必须在 512～2048 位之间。diffie hellman 密钥必须在 128～4096 位之间。dsa 密钥必须在 512～1024 位之间，而且必须是 64 的整数倍。hmac-md5 密钥必须在 1～512 位之间。
- -n nametype：指定密钥的所有者类型。nametype 的值必须是 zone（针对 DNSSEC 区域密钥）、host 或 entity（针对主机关联密钥）、user（针对用户关联密钥）。这些值不区分大小写。
- -c class：指定包含密钥的 DNS 记录应该有指定类。如果未指定，则使用类 in。

- -e：如果生成 rsa 密钥，则使用大指数。
- -g generator：如果生成 diffie hellman 密钥，则使用该发生器。允许值为 2 和 5。如果不指定生成器，可能的话使用来自于 RFC 2539 的已知质数；否则默认值为 2。
- -h：打印 dnssec-keygen 的选项和参数的简短摘要。
- -pprotocol：为生成的密钥设定协议值。协议为 0～255 之间的某个数。类型 user 的默认值是 2（E-mail），其他类型的默认值是 3（DNSSEC）。该参数的其他可能值列在 RFC 2535 和它的后续作业中。
- -r randomdev：指定随机源。如果操作系统不提供/dev/random 或等价设备，默认随机源为键盘输入。randomdev 指定包含要使用随机数据的字符设备名或文件名，而不是默认值。特殊值键盘指示应该使用键盘输入。
- -s strength：指定密钥的强度值。该强度为 0～15 之间的某个数，目前在 DNSSEC 中尚无定义用途。
- -t type：指定密钥的使用。类型必须是 authconf、noauthconf、noauth 或 noconf 之一。默认值是 authconf。auth 指验证数据的能力，conf 指加密数据的能力。
- -v level：设置调试级别。

说明如下。

当 dnssec-keygen 成功完成时，它显示如 knnnn.+aaa+iiiii 这样的字符串到标准输出。这是生成的密钥的标识字符串，这些字符串可作为 dnssec-makekeyset 的参数。

- nnnn：密钥名。
- aaa：算法的数字表示。
- iiiii：密钥标识符（或占地面积）。
- dnssec-keygen：创建两个名称基于所显示字符串的文件。knnnn.+aaa+iiiii.key 包含公用密钥，而 knnnn.+aaa+iiiii.private 包含专用密钥。.key 文件包含一个 DNS 密钥记录，它可以插入到一个区域文件中（直接或使用$include 语句）。.private 文件包含算法特定字段。出于安全性原因，该文件一般没有读许可权。.key 和.private 文件由诸如 hmac-md5 等对称加密算法创建，即使公共密钥和专用密钥相同。

应用实例如下。

要为域 example.com 生成 768bit 的 dsa 密钥，请输入下列命令：

```
#dnssec-keygen -a dsa -b 768 -n zone example.com
```

此命令显示如下用法的字符串：

```
kexample.com.+003+26160
```

在这个例子中，dnssec-keygen 创建了文件 kexample.com.+003+26160.key 和 kexample. com.+003+26160.private。

9.2.9 dnssec-signkey：DNSSEC 密钥集签名工具

作用：dnssec-signkey 命令签署一个密钥集。典型的密钥集是子区域的，并且将由

dnssec-makekeyset 创建。子区域的密钥集由父区域用区域键签名。输出文件的用法为 signedkey-nnnn，其中 nnnn 为区域名称。

用法：dnssec-signkey [选项] keyset key

主要选项如下。

- -a：验证所有生成的签名。
- -c class：指定密钥集的 DNS 类。
- -s start-time：在生成的 SIG 记录变为有效时，指定日期和时间。可以是绝对或相对时间。绝对启动时间的用法为 YYYYMMDDHHMMSS 数字符号，如 20000530144500 表示 2000 年 5 月 30 日 14:45:00 UTC。相对启动时间用+N 标记，其中 *N* 表示离现在的秒数。如果 start-time 未指定，则使用当前时间。
- -e end-time：在生成的 SIG 记录到期时，指定日期和时间。若带有启动时间，绝对时间的用法为 YYYYMMDDHHMMSS 符号。相对启动时间用+N 标记，表示距启动时间 *N* 秒。相对当前时间的时间记录用 now+N 标记。如果未指定 end-time，默认值为从启动时间算起 30 天。
- -h：显示 dnssec-signkey 的选项和参数的简短摘要。
- -p：签署区域时使用伪随机数据。这比使用真正的随机数据快，但安全性差。这个选项在标记大区域或者熵源有限时非常有用。
- -r randomdev：指定随机源。如果操作系统不提供/dev/random 或类似设备，默认随机源为键盘输入。randomdev 指定包含随机数的字符设备名或文件名，而不是默认值。特定的键盘值指定使用键盘输入。
- -v level：设置调试级别。
- keyset：包含子密钥集的文件。
- key：用于子密钥集签名的密钥。

应用实例如下。

DNSSEC 识别的.com 区域的 DNS 管理员将使用以下命令来签署由 dnssec-makekeyset 带 dnssec-keygen 生成的密钥创建的 example.com。

```
#dnssec-signkey keyset-example.com. Kcom.+003+51944
```

在这个例子中，dnssec-signkey 创建文件 signedkey-example.com，它包含 example.com 密钥和.com 密钥生成的签名。

9.2.10　dnssec-makekeyset：DNSSEC 区域签名工具

作用：dnssec-makekeyset 命令生成 dnssec-keygen 创建的一个或多个密钥组成的密钥集。它创建一个文件，包含针对每个密钥的密钥记录，用每个区域密钥自签署密钥集。输出文件的形式为 keyset-nnnn，其中 nnnn 是区域名。

用法：dnssec-makekeyset [选项] [key]

主要选项如下。

- -a：验证所有生成的签名。

- -s start-time：在生成的 SIG 记录合法化时，指定日期和时间。可以是绝对或相对时间。绝对起始时间的用法为 YYYYMMDDHHMMSS 数字符号，如 20000530144500 表示 2000 年 5 月 30 日 14:45:00 UTC。相对起始时间用+N 指示，表示截至现在的秒数为 *N* 秒。如果 start-time 未指定，则使用当前时间。
- -e end-time：在生成的 SIG 记录到期时，指定日期和时间。若指定 start-time，则绝对时间的用法为 YYYYMMDDHHMMSS。相对于启动时间的时间用+N 标记，表示距启动时间 *N* 秒。相对当前时间的时间用 now+N 标记。如果未指定 end-time，默认值为从启动时间算起 30 天。
- -h：显示 dnssec-makekeyset 的选项和参数的简短摘要。
- -p：签署区域时使用伪随机数据。这比使用真正的随机数据快，但安全性差。这个选项在标记大区域或者熵源有限时非常有用。
- -r randomdev：指定随机源。如果操作系统不提供/dev/random 或类似设备，默认随机源为键盘输入。randomdev 指定包含要使用的随机数据的字符设备名或文件名，而不是默认值。特定的键盘值指定使用键盘输入。
- -t ttl：指定记录 KEY 和 SIG 的生存时间。默认是 3600 秒。
- -v level：设置调试级别。
- Key：密钥集文件要包含的密钥列表。这些密钥的用法为 Knnnn.+aaa+iiiii，由 dnssec-keygen 生成。

应用实例如下。

下面的命令针对 example.com 生成一个包含 DSA 密钥的密钥集。

```
    #dnssec-makekeyset -t 86400 -s 20000701120000 -e +2592000
Kexample.com.+003+26160
```

在这个例子中，dnssec-makekeyset 创建文件 keyset-example.com，该文件包含指定密钥和自建签名。example.com 的 DNS 管理器可以发送 keyset-example.com 给.com 的 DNS 管理器要求签名，如果.com 区域是 DNSSEC 识别的，而且两个区域的管理器拥有相互认证、安全的进行密钥和签名交换的机制。

9.2.11 dnssec-signzone：DNSSEC 区域签名工具

作用：dnssec-signzone 命令签署一个区域。它生成 NXT 和 SIG 记录，并产生区域的已签署的版本。如果父区域有文件 signedkey，则父域的签名将合并到生成的已签署区域文件。已签署区域的授权的安全状态（即无论子域是否安全）由每个子域是否含有 signedkey 文件决定。

用法：dnssec-signzone[选项]zonefile key...

主要选项如下。

- -a：验证所有生成的签名。
- -c class：指定区域的 DNS 类。
- -d directory：作为目录在目录中查找 signedkey 文件。
- -s start-time：在生成的 SIG 记录变为有效时，指定日期和时间。可以是绝对或相对时间。

绝对启动时间的用法为 YYYYMMDDHHMMSS 数字符号，如 20000530144500 表示 2000 年 5 月 30 日 14:45:00 UTC。相对启动时间用+N 标记，表示距当前时间 *N* 秒。如果 start-time 未指定，则使用当前时间。

- -e end-time：在生成的 SIG 记录到期时，指定日期和时间。若带有启动时间，绝对时间的用法为 YYYYMMDDHHMMSS 符号。相对启动时间用+N 标记，表示距启动时间 *N* 秒。相对当前时间的时间记录用 now+N 标记。如果未指定 end-time，则默认值为从启动时间算起 30 天。
- -f output-file：包含签名区域的输出文件名。默认为送到输入文件的 append.signed。
- -h：显示 dnssec-signzone 的选项和参数的简短摘要。
- -i interval：当前一个签名区域作为输入传入时，记录被重新签名。时间间隔选项以距离目前时间的偏移量（秒数）指定间隔周期。如果在间隔周期后记录 SIG 过期了，则保留它；否则，认为它马上会过期并替换它。默认间隔周期为签名启动时间和结束时间间隔的 1/4。因此，如果未指定 end-time 和 start-time，dnssec-signzone 将生成合法期为 30 天、间隔周期为 7.5 天的签名。因此，任何存在的记录 SIG 如果将在 7.5 天内过期，则会被替换。
- -n ncpus：指定使用的线程数。默认情况下每个检测到的 CPU 启动一个线程。
- -o origin：区域原点。若不指定，则假定区域文件的名称为原点。
- -p：签署区域时使用伪随机数据。这比使用真正的随机数据快，但安全性差。这个选项在标记大区域或者熵源有限时非常有用。
- -t：完成时显示统计信息。
- -v level：设置调试级别。
- -r randomdev：指定随机源。如果操作系统不提供/dev/random 或类似设备，默认随机源为键盘输入。randomdev 指定包含随机数的字符设备名或文件名，而不是默认值。特定的键盘值指定使用键盘输入。
- zonefile：包含需被签名的区域的文件。设置调试级别。
- key：用于签署子密钥集的密钥。

应用实例如下。

生成 DSA 密钥签署 example.com 区域。区域密钥必须在区域中。如果有与本区域或任何子域关联的 signedkey 文件，它们必须在当前目录 example.com 发出以下命令：

```
#dnssec-signzone -o example.com db.example.com Kexample.com.+003+26160
```

在这个例子中，dnssec-signzone 创建文件 db.example.com.signed。该文件必须在文件 named.conf 的区域语句中引用。

9.2.12　dlint：Bind DNS 服务器辅助工具

除了系统安装产生的工具（nslookup、dig、host、named-checkzone、named-checkconf、rndc）外，这里再介绍一个开放源代码工具：dlint。一个配置不完善的 DNS 服务器存在很大的安全漏洞，dlint 可以帮助分析 DNS 配置文件中的问题。dlint 是一个专门检查 DNS 配置文件的开放

源代码软件，若要运行它，系统要求安装 Perl 语言和 dig 命令。

```
#rpm –qa|grep perl
#dig hostname any
perl 5.8.0
; <<>> dig 9.2.1 <<>>
;; global options:printcmd
```

软件下载安装：

```
#wget http://www.domtools.com/pub/dlint1.4.0.tar.gz
#gunzip dlint1.4.0.tar.gz
#tar vxf dlint1.4.0.tar;#cd dlint1.4.0
#make install
```

系统会将 dlint 安装在/usr/bin/目录下，并执行如下步骤。

① dlint 首先检查配置文件的拼写错误。

② 有 a（address）记录的主机名称必须有 ptr（称 pointer，ptr 反向解析）记录。如果有 a 记录的主机名称没有 ptr，则配置文件不能通过。dlint 可以在用户配置文件中为 a 记录查找丢失的 ptr 记录。

③ dint 会记录 in-addr.arpa 区带的每一条 ptr 记录是否有对应的 a 记录存在。

④ dint 会以递归的方式检查子区带，查找它们的配置问题。dlint 软件可以分析任何 DNS 区带。

⑤ dlint 软件包的主要脚本是 dlint，使用方法如下：

```
#/usr/bin/dlint domain
```

9.2.13　dnstop：Bind DNS 服务器辅助工具

在维护 DNS 服务器时，网管员希望知道到底是哪些用户在使用 DNS 服务器，同时也希望能对 DNS 状态查询做一个统计。可以使用 dnstop 查询 DNS 服务器状态。

软件下载和安装：

```
#wegt http://www.stearns.org/dnstop/dnstop-20040309-1.i386.rpm
#rpm –ivh dnstop-20040309-1.i386.rpm
```

运行软件。如果想查看通过 eth0 的 DNS 网络流量，可以使用如下命令：

```
#dnstop -s eth0
0 new queries, 6 total queries             mon jul6 09:54:35 2008
sources              count%
---------------- --------- ------
192.168.0.117           4       66.7
192.168.0.143           1       16.7
192.168.0.42            1       16.7
```

在运行 dnstop 的过程中，可以输入 s,d,t,1,2,3,ctrl+r,ctrl+x 命令，以交互方式来显示不同的信息。

- 1：（tld）记录查询的顶级域名。
- 2：（sld）记录查询的二级域名。
- 3：（sld）记录查询的三级域名。
- s：（source）记录发送 DNS 查询的客户端 IP 地址表。
- d：（destinations）记录 DNS 查询的目的服务器的 IP 地址表。
- t：（query type）记录查询详细类型。
- ctrl+r：重新记录。
- ctrl+x：退出。

9.3 NFS 服务器管理命令

9.3.1　nfsd：启动、停止 NFS 服务器

```
# service nfs start
```

应用实例如下。

```
启动 NFS 服务：                                    [确定]
启动 NFS 守护进程：                                [确定]
启动 NFS mountd：                                  [确定]
正在启动 RPC idmapd：                              [确定]
```

> **说明**
> 运行以上命令前要运行 portmap 命令。

9.3.2　portmap：将 RPC 程序号转换为因特网端口号

作用：portmap 守护程序用于将 RPC 程序号转换为因特网端口号。

用法：portmap [-dvl]

主要选项如下。

- -d：调试状态下运行 portmap。
- -l：只监听 localhost 端口。
- -v：运行 portmap 使用详细描述。

应用实例如下。

运行 portmap。

```
#portmap
```

或者使用如下命令：

```
/etc/rc.d/init.d/portmap start
```

说明

当 RPC 服务器启动时，它向 portmap 守护程序注册。服务器告诉守护程序它正在监听哪个端口号，以及它服务于哪个 RPC 程序号。这样，portmap 守护程序就知道主机上每个已注册端口的位置和每个端口上哪个程序可用。对每一个客户机试图调用的程序，客户机仅咨询 portmap 守护程序一次。portmap 守护程序告诉客户机该调用被发送到哪个端口。客户机存储该信息以备将来引用。既然标准 RPC 服务器通常被 xinetd 守护程序启动,portmap 守护程序必须在 xinetd 守护程序被调用之前启动。如果 portmap 守护程序被终止或异常结束，主机上的所有 RPC 服务器必须重新启动。

9.3.3 rpcinfo：报告远程过程调用（RPC）服务器的状态

作用：rpcinfo 命令造成 RPC 调用到 RPC 服务器并报告服务器状态。

用法：rpcinfo [选项]

主要选项如下。

- -b：将 RPC 广播到指定的程序号和版本号的过程 0 并报告响应的所有主机。如果指定传输，它仅广播指定传输上的请求。如果任何传输都不支持广播，则打印错误消息。由于可能对其他系统产生不利影响，应该限制使用广播（-b 标志）。
- -d：删除指定程序号和版本号的 RPC 服务的注册。如果使用传输，只注销那个传输上的服务；否则注销所有注册的传输上的服务。本选项仅由 root 用户执行。
- -l：显示在指定主机上的指定程序号和版本号的条目的列表，返回所有使用与用于联系远程 portmap 守护程序的协议在同一个协议系列的传输的条目。本标志只适用于 AIX 4.2.1 或更高版本。
- -m：显示指定主机上的端口映象操作统计信息表。此表包含对每个版本的端口映象（版本 2、3 和 4）的统计信息、每个过程请求和成功服务的次数、所做的远程调用请求的类型和数目，以及关于已处理的 RPC 地址查询的信息。本信息用来监视在主机上的 RPC 活动。本标志只适用于 AIX 4.2.1 或更高版本。
- -n port：使用 port 参数作为-t 和-u 选项的端口号来替换端口映射给出的端口号。使用-n 选项来避免到远程端口映象查找服务地址的调用。使用-a 选项废弃本选项。
- -p：探测使用端口映射协议版本 2 的主机上的 portmap 服务，并显示所有已注册的 RPC 程序的列表。如果不指定主机，默认为本地主机。
- -s：显示主机上所有已注册的 RPC 程序的简明列表。如果不指定主机，默认为本地主机。

本标志只适用于 AIX 4.2.1 或更高版本。

- -t：执行一个到使用 TCP 的指定主机上的程序号的过程 0 的 RPC 调用，并报告是否收到响应。当使用在第三条语法中显示的-t 选项时废弃本选项。
- -u：执行一个到使用 UDP 的指定主机上的程序号的过程 0 的 RPC 调用，并报告是否收到响应。当使用在第三条语法中显示的-t 选项时废弃本选项。

应用实例如下。

（1）显示在本地机器上注册的所有 RPC 服务。

```
#rpcinfo-p
```

（2）显示在指定机器上注册的所有 RPC 服务。

```
#rpcinfo-p zelda
```

在本例中，rpcinfo 显示在名为 zelda 的机器上注册的所有 RPC 服务。

（3）显示本地网络上运行指定服务器的某个版本的所有机器。

```
#rpcinfo-b ypserv 2
```

在本例中，rpcinfo 命令显示运行 ypserv 守护程序的版本 2 的所有机器的列表。

（4）删除服务注册。

```
rpcinfo-d sprayd 1
```

在本例中，rpcinfo 命令删除 sprayd 守护程序的版本 1。

说明

rpcinfo 命令造成 rpRPC 调用到 RPC 服务器，并报告服务器状态。例如，本命令报告服务器是否就绪并处于等待中或不可用。程序参数可以为名称或号码。如果指定版本，rpcinfo 命令试图调用指定程序的那个版本；否则，rpcinfo 命令试图查找所有通过调用版本 0（零）来指定的程序的已注册版本号，然后试图调用所有已注册的版本（版本 0 假定为不存在。如果确实存在，rpcinfo 命令将试图通过调用非常高的版本号来获得此信息）。

9.3.4　showmount：显示远程已安装文件系统的所有客户机的列表

作用：showmount 命令可显示以下内容之一。

- 已经远程挂载了通过 NFS 服务器共享的文件系统的所有客户机。
- 仅由客户机挂载的文件系统。
- 共享的文件系统及客户机访问信息。

用法：showmount [-adehv] [--no-headers]

主要选项如下。

- -a 或--all：以 host:dir 这样的用法来显示客户主机名和挂载点目录。

- -d 或--directories：仅显示被客户挂载的目录名。
- -e 或--exports：显示 NFS 服务器的输出清单。
- -h 或--help：显示帮助信息。
- -v 或--version：显示版本信息。
- --no-headers：禁止输出描述头部信息。

应用实例如下。

（1）显示主机安装的所有远程目录的列表。

```
#/usr/bin/showmount-a zeus
```

在本例中，showmount 命令产生名为 zeus 的主机上客户机安装的所有远程目录的列表。

（2）显示主机上的客户机安装的目录列表。

```
#/usr/bin/showmount-d athena
```

在本例中，showmount 命令产生名为 athena 的主机上客户机安装的所有远程目录的列表。

（3）打印机器上导出的所有目录列表。

```
#/usr/bin/showmount-e zeus
```

在本例中，showmount 命令产生名为 zeus 的主机导出的所有远程目录的列表。

9.3.5 umount：删除当前已挂载的远程文件系统

作用：使用 umount 命令，可以删除当前已挂载的远程文件系统。

用法：umount [-f] [-r] [-n] [-v] node...

主要选项如下。

- -a：卸载所有安装的文件系统。
- -f：在远程环境中进行强行卸载。用于在服务器当机或服务器路径名无法解析，或必须将使用中的文件系统卸载时释放客户机。-f 标志不支持日志文件系统。
- -n node：指定要卸载的已安装目录占用的节点。umount -n node 命令卸载所有用 node 参数进行的远程安装。
- -t type：卸载包含 type=type 标志并已安装的/etc/filesystems 文件中的所有节。type 参数是字符串值，如指定组名的远程值。
- -v：只进行测试。

应用实例如下。

（1）从远程节点 node a 卸载所有安装。

```
#umount-n nodea
```

（2）卸载特定类型的文件和目录。

```
#umount-t test
```

这样便卸载了在包含 type=test 属性的/etc/filesystems 文件中有一个节点的所有文件或目录。

9.3.6　mount：将已命名的文件系统连接到指定的挂载点

作用：使用 mount 命令，可以将已命名的文件系统（本地或远程的）连接到指定的挂载点。有关更多信息，请参见 mount(1m)手册页。在不使用参数的情况下，mount 将显示当前在计算机上挂载的文件系统列表。

用法：mount[NFS 选项]

主要 NFS 选项如下。

- bg|fg：可以使用这些选项来选择挂载失败时的重试行为。bg 选项将导致挂载试图在后台运行；fg 选项将导致挂载试图在前台运行。默认值为 fg，对于必须可用的文件系统而言，这是最佳选择。此选项可防止在挂载完成之前进一步进行处理。bg 是适用于非关键文件系统的适当选择，因为客户机在等待挂载请求完成的同时可以执行其他处理。
- public：此选项强制在与 NFS 服务器联系时使用公共文件句柄。如果服务器支持公共文件句柄，则由于不使用 mount 协议，因此挂载操作会比较快。此外，由于不使用 mount 协议，公共选项允许通过防火墙进行挂载。
- rw|ro：-rw 和-ro 选项指示文件系统以读/写方式挂载还是以只读方式挂载。默认值为读/写方式，该选项适用于远程起始目录、邮件假脱机目录和需要由用户更改的其他文件系统。只读选项适用于不应该由用户更改的目录。例如，用户不应写入手册页的共享副本。
- soft|hard：如果服务器没有做出响应，则使用 soft 选项挂载的 NFS 文件系统将返回错误。hard 选项将导致挂载继续重试，直到服务器做出响应为止。默认值为 hard，大多数文件系统都应该使用此选项。应用程序不会经常检查从使用 soft 选项挂载的文件系统返回的值，这可能会使应用程序出现故障或可能导致文件损坏。如果应用程序检查返回值，则路由问题和其他情况可能仍然会干扰应用程序或导致文件损坏（如果选择 soft 选项）。在大多数情况下，不应该使用 soft 选项。如果文件系统是使用 hard 选项挂载的且不可用，则使用该文件系统的应用程序将挂起，直到该文件系统可用为止。

应用实例如下。

（1）以只读方式从服务器 bee 挂载 NFS 文件系统。

```
# mount -f nfs -r bee:/export/share/man /usr/man
```

（2）使用客户机故障转移。

```
# mount -f nfs -r bee,wasp:/export/share/man /usr/man
```

9.3.7　automount：安装自动安装点

作用：automount 命令用作 autofs 的一个管理工具。

用法：automount [选项]

主要选项如下。

- -t duration：用于设置文件系统持续处于挂载状态的时间（以秒为单位）。
- -v：用于选择详细模式。在详细模式下运行此命令可以更容易地排除故障。

应用实例如下。

在标准输出上显示详细状态和警告消息。

```
#automount -v
```

9.3.8 mountall：挂载一组文件系统

作用：使用mountall命令可挂载文件系统表中列出的所有文件系统或特定的一组文件系统。

用法：mountall [选项]

主要选项如下。

- -f fstype：选择要访问的文件系统类型。
- -r：选择文件系统表中列出的所有远程文件系统。
- -l：选择所有本地文件系统。

应用实例如下。

（1）卸装所有从 host.domain 加载的不可卸装的文件系统。

```
#umountall -h host.domain
```

（2）卸装所有远程加载的文件系统。

```
#umountall -r
```

9.3.9 exportfs：重新分享/etc/exports 变更的目录资源

作用：修改了/etc/exports 文件后，不需要重新激活 NFS，只需重新扫描一次/etc/exports 的文件，并且重新将设定加载即可。

用法：exportfs [-aruv]

主要选项如下。

- -a：全部挂载（或卸载）/etc/exports 文件内的设定。
- -r：重新挂载/etc/exports 里的设定。此外，也同步更新/etc/exports 及/var/lib/nfs/xtab 的内容。
- -u：卸载某一目录。
- -v：在 export 的时候，将共享的目录显示到屏幕上。

应用实例如下。

（1）导出/etc/exports 文件中的全部目录。

```
#exportfs -a
```

（2）从/etc/exports 文件导出一个目录。

```
#exportfs /home/notes
```

在这个例子中，/home/notes 目录被导出。要让这个命令工作，必须在/etc/exports 文件中指定/home/notes 目录。

（3）取消导出目录。

```
#exportfs -u /home/notes
```

在这个例子中，/home/notes 目录不导出。

（4）显示当前正在导出的目录。

```
#exportfs -v
```

9.4 Samba 服务器管理命令

9.4.1　启动 Samba 服务器

可以执行下面的命令来启动 Samba 服务器：

```
# service smb start
```

一旦能看到下面的显示，就表示启动成功。

```
startingsmbservice[确定]
startingnmbservice[确定]
```

如果要暂停或重新启动 Samba 服务器，只要将上述命令中的 start 改为 stop 或 restart 就可以了。

9.4.2　testparm：检查 smb.conf 配置文件的内部正确性

作用：testparm 用于检查 smb.conf 配置文件的内部正确性，此程序是 Samba 套件的一部分。

用法：testparm [-s] [-h] [-l servername] [configfilename] [hostname hostip]

主要选项如下。

- -s：如果不带这个参数，testparm 将提示在列出服务名和服务定义项之间按一下 Enter 键。
- -h：列出用法信息。
- -l servername：对服务项名字设定%l 这样的宏值。对于测试包含这样的宏值的文件非常有用。
- configfilename：指定要检查的配置文件名。如果不指定的话，程序将对默认的 smb.conf 文件进行检查。
- hostname：如果命令行带有这个参数时，测试程序将检查 smb.conf 文件中的“hosts allow”和“hosts deny”参数，用以测试这些 IP 地址对应的主机名是否可以访问 smbd 服务器。选用这样的参数时，hostip 参数也必须一并使用。
- hostip：这个参数用于指定前面给出的主机名相对应的 IP 地址。如上所述，主机名参数和这个地址必须一并使用。

应用实例如下。

测试 Samba 配置文件。设置好 smb.conf 文件之后，执行下面的命令测试 smb.conf 文件的设置语法是否正确。

```
#testparm
load smb config files from /etc/samba/smb.conf
processing section "[homes]"
processing section "[printers]"
processing section "[tmp]"
processing section "[public]"
loaded services file ok.
press enter to see a dump of your service definitions
```

上述结果表示正确。检查正常后，一定要记得执行如下命令：

```
# service smbrestart
```

即重新启动 Samba 服务器。

说明

testparm 是一个非常简单的程序，用于检查 smbd 配置文件的内部正确性。如果程序报告没有什么问题，那么就可以放心地使用这个配置文件，并且 smbd 也会非常顺利地装入它。

9.4.3 smbd(samba daemon)：Samba 服务器程序

作用：smbd 为 Samba 服务器程序，可分享文件与打印机等网络资源供 Windows 相关的用户端程序存取。

用法：smbd [选项] [-d<排错层级>] [-i<范围>] [-l<记录文件>] [-o<连接槽选项>] [-p<连接端口号>] [-s<配置文件>]

主要选项如下。

- -a：所有的连线记录都会加到记录文件中。
- -d<排错层级>：指定记录文件所记载事件的详细程度。
- -d：使用此参数时，smbd 会以服务程序的方式在后台执行。
- -h：显示帮助。
- -i<范围>：指定 netbios 名称的范围。
- -l<记录文件>：指定记录文件的名称。
- -o：每次启动时，会覆盖原有的记录文件。
- -o<连接槽选项>：设置连接槽选项。
- -p<连接端口编号>：设置连接端口编号。这个端口号用于从客户端建立对服务器的连接。

基于 TCP 上的 SMB 所用的标准（通常使用的）端口号为 139，因此这是默认值。

- -p：仅用来测试 smbd 程序的正确性。
- -s<配置文件>：指定 smbd 的设置文件。

应用实例如下。

显示帮助。

```
#smbd –h
```

9.4.4　smbclient：类似 FTP 操作方式的访问 SMB/CIFS 服务器资源的客户端

作用：SMB 与 CIFS 为服务器通信协议，常用于 Windows 系统。smbclient 命令可让 Linux 系统存取 Windows 系统所分享的资源。

用法：smbclient [网络资源] [密码] [-ehln] [-b<IP 地址>] [-d<排错层级>] [-i<范围>] [-i<IP 地址>] [-l<记录文件>] [-m<NetBIOS 名称>] [-n<NetBIOS 名称>] [-o<连接槽选项>] [-p<TCP 连接端口>] [-r<名称解析顺序>] [-s<目录>] [-t<服务器字码>] [-t<tar 选项>] [-u<用户名称>] [-w<工作群组>]

主要选项如下。

- [网络资源]：[网络资源]的用法为“//服务器名称/资源分享名称”。
- [密码]：输入存取网络资源所需的密码。
- -b<IP 地址>：传送广播数据包时所用的 IP 地址。
- -d<排错层级>：指定记录文件所记载事件的详细程度。
- -e：将信息送到标准错误输出设备。
- -h：显示帮助。
- -i<范围>：设置 NetBIOS 名称范围。
- -i<IP 地址>：指定服务器的 IP 地址。
- -l<记录文件>：指定记录文件的名称。
- -l：显示服务器端所分享出来的所有资源。
- -m<NetBIOS 名称>：可利用 WinPopup 协议，将信息送给选项中所指定的主机。
- -n<NetBIOS 名称>：指定用户端所要使用的 NetBIOS 名称。
- -n：不用询问密码。
- -o<连接槽选项>：设置用户端 TCP 连接槽的选项。
- -p<TCP 连接端口>：指定服务器端 TCP 连接端口编号。
- -r<名称解析顺序>：设置 NetBIOS 名称解析的顺序。
- -s<目录>：指定 smb.conf 所在的目录。
- -t<服务器字码>：设置用何种字符码来解析服务器端的文件名称。
- -t<tar 选项>：备份服务器端分享的全部文件，并打包成 TAR 格式的文件。
- -u<用户名称>：指定用户名称。

– -w<工作群组>：指定工作群组名称。

说明

此程序是 Samba 套件的一部分。smbclient 是一个可以和 SMB/CIFS 服务器“交谈”的客户端程序。它提供了类似 FTP 程序的用户界面。它可以完成的操作包括从服务器下载文件到本地、上传本地文件到服务器及在服务器上查找目录信息等。

应用实例如下。

（1）在 Linux 下使用下面的命令检查服务器所共享的资源。

```
#smbclient -l localhost
added interface ip=192.168.0.11 bcast=192.168.1.255 nmask=255.255.255.0
password:
domain=[workgroup] os=[unix] server=[3.0.10-1.4e]
       sharename     type     comment
       ---------     ----     -------
       root director  disk
       root          disk
       ipc$          ipc      ipc service (samba server)
       admin$        disk     ipc service (samba server)

       server              comment
       ---------           -------
       host                samba server
       workgroup           master
       ---------           -------
       workgroup
```

（2）以 cjh 用户身份访问 win01 计算机的 tools 共享。

```
# smbclient //win01/tools -u cjh
added interface ip=192.168.1.200 bcast=192.168.1.255
nmask=255.255.255.0
// 输入 Windows 用户 cjh 的密码
password:
domain=[cjh1] os=[windows 5.0] server=[windows 2000 lan manager]
//显示 smbclient 可用的子命令
smb: \> ?
? altname archive blocksize cancel
```

```
cd chmod chown del dir
du exit get help history
lcd link lowercase ls mask
md mget mkdir more mput
newer open print printmode prompt
put pwd q queue quit
rd recurse rename rm rmdir
setmode symlink tar tarmode translate
!
//接下来就可以像使用 FTP 客户端一样来使用 smbclient
smb: \> quit
#
```

9.4.5　smbstatus：报告当前 Samba 的连接状态

作用：此程序是 Samba 套件的一部分。smbstatus 是一个非常简单的程序，用于列出当前 Samba 的连接状态。

用法：smbstatus [-b] [-d] [-l] [-p] [-s] [-s configuration file] [-u username]

主要选项如下。

- -b：指定只输出简短的内容。
- -d：指定以详细方式输出内容。
- -l：让 smbstatus 只列出/var 目录中的被锁定项。
- -p：列出 smbd 进程的列表然后退出。该参数对脚本编程很有用。
- -s：让 smbstatus 只列出共享资源项。
- -sconfigurationfile：指定一个配置文件。当然，在编译时已经做好了默认的配置文件，文件中包含了服务所需的详细配置信息。参见 smb.conf（5）获得更多信息。
- -uusername：查看只与 username 用户对应的信息。

应用实例如下。

（1）在 Linux 下使用下面的命令查看 Samba 服务器资源被使用的情况。

```
#smbstatus
samba version 3.0.10-1.4e
pid    username     group       machine
-------------------------------------------------------------------
 4231   adm          adm          x-41        (192.168.0.11)

service     pid    machine      connected at
-------------------------------------------------------
ipc$        4231   x-41         sun jul 16 01:13:14 2006
```

```
adm          4231   x-41         sun jul 16 01:16:52 2006
no locked files
```

（2）查看简要的使用信息。

```
# smbstatus -b
samba version3.0.23c-2
pid username machine time logged in
-------------------------------------------------------------
7485 lrj win01 sat may 24 01:42:38 2003
7476 lrj win01 sat may 24 01:01:36 2003
```

9.4.6　smbmount：装载一个 smbfs 文件系统

作用：smbmount 可以装载一个 Linux SMB 文件系统。

用法：smbmount {service} {mount-point} [选项]

主要选项如下。

- username=<arg>：指定连接时的用户名。如果不指定，程序会用 user 这个环境变量。这个选项也接受 user%password、user/workgroup 或 user/workgroup%password 形式的参数，允许密码和工作组作为用户名的一部分同时被指定。
- password=<arg>：指定 SMB 密码。如果不指定，将使用环境变量 passwd。如果没有指定，也没有环境变量，smbmount 会提示输入密码，除非使用了 guest 选项。
- krb：使用 Kerberos 协议。
- netbiosname=<arg>：设定源 NetBIOS 主机名。默认是本机名。
- gid=<arg>：设定装载的文件系统的组标识符。可以用组名或者 GID 数字两种方法来指定。
- port=<arg>：设定远程 SMB 系统的端口号。默认是 139。
- fmask=<arg>：设定文件掩码。这决定了本地文件系统中远程文件的权限。它不是掩码，而是实际的文件权限。默认值是根据当前的 umask 算出来的。
- dmask=<arg>：设定目录掩码。这决定了本地文件系统中远程目录的权限。它不是掩码，而是实际的目录权限。默认值是根据当前的 umask 算出来的。
- debug=<arg>：设定调试等级。对于跟踪 SMB 的连接问题非常有用。建议的值是 4。如果设置得过高，会产生大量输出，可能会掩盖有用的输出。
- ip=<arg>：设定目标主机或 IP 地址。
- workgroup=<arg>：设定目的工作组。
- sockopt=<arg>：设定 TCP 套接字的一些选项。参见 smb.conf（5）配置文件中的 socketoptions 选项。
- scope=<arg>：设定 NetBIOS 范围。
- guest：不提示密码。
- ro：以只读方式装载。
- rw：以读/写方式装载。

> **说明**
>
> 在 Linux 环境下使用共享资源的另一种方法是使用远程挂载方法将远程共享挂载到本地，这类似于在 Windows 环境下映射网络驱动器。使用远程挂载方法访问共享需要使用 Smaba 系统支持。

应用实例如下。

使用 smbmount 命令挂载远程共享。

```
//创建挂载点目录
# mkdir –p /mnt/smb/win01
//将远程共享 tools 挂载到本地/mnt/smb/win01 目录
# smbmount //win01/tools /mnt/smb/win01
//接下来就可以像使用本地文件系统一样使用共享资源了
```

9.4.7　smbpasswd：设置用户的 SMB 密码

作用：此程序是 Samba 套件的一部分，用来设置用户的 SMB 密码。

用法：smbpasswd [选项]

主要选项如下。

- -a：在这个选项后跟用户名，用来实现在本地 smbpasswd 文件中增加用户，并且同时设置密码（提示原密码时用<enter>）。如果 smbpasswd 文件中已经存在了这个用户，命令就变成通常的修改密码模式。注意，默认的 passdb 后端要求所要加入的 SMB 用户必须是系统密码文件中（通常是/etc/passwd）已经存在的用户，否则加入操作将会失败。只有 root 运行 smbpasswd 程序时才可以使用这个选项。
- -d：这个选项后跟用户名，用来禁止存在于 smbpasswd 文件中的这个账号。通过在 smbpasswd 文件的账号控制部分写入“d”标志来实现这个功能。一旦账号被禁止，所有使用这个账号进行 SMB 身份验证的尝试都将失败。只有 root 运行 smbpasswd 程序时才可以使用这个选项。
- -e：这个选项后跟用户名，用来在本地 smbpasswd 文件中的这个账号被禁止时重新允许使用。如果账号并未被禁止，使用这个选项不会有什么结果。被允许的账号将可以通过 SMB 的身份验证。只有 root 运行 smbpasswd 程序时才可以使用这个选项。
- -d：debugleveldebuglevel 是一个 0～10 的整数。如果没有指定此参数，则默认值是 0。这个值越大，就会有越多关于 smbpasswd 的详细活动信息将被记录到文件中。在 0 调试级时，只记录紧急错误和严重警告。
- -n：用这个选项后跟用户名来把这个账号的密码设为空（也就是没有密码）。程序会把本地 smbpasswd 文件中该密码项的第一部分改为“nopassword”。只有 root 运行 smbpasswd 程序时才可以使用这个选项。
- -r remotemachinename：使用这个选项来让用户指定他们所希望改变密码的主机，不用

此选项时默认对本地更改密码。SMB/CIFS 服务器会试图连接以 remote machinename 作为 NetBIOS 名字的主机以更改密码。

- -rnameresolveorder：用这个选项来让使用 smbclient 的用户在查询主机 NetBIOS 名字用于连接时，决定使用怎样的名字解析服务。这些名字解析选项是 lmhosts、host、wins 和 bcast。
- -m：这个选项用来把账号改为一个 machine 账号。通常当 Samba 作为 Windows NT 主域控制器时可以使用它。只有 root 运行 smbpasswd 程序时才可以使用这个选项。
- -u username：这个选项只能和-r 选项联合使用。当从远程主机更改密码时，用它来允许用户指定要改变的远程主机密码的用户账号。这使得在不同的系统上使用不同账号的用户可以修改密码。
- -h：使用这个选项可以打印出 smbpasswd 的帮助信息，root 用户和普通用户都可以使用。
- -s：使用这个选项会使 smbpasswd 保持安静（不发出提示），在标准输入上读取原密码和新密码，而不是从/dev/tty 上读取密码（像 passwd 那样）。使用脚本来处理 smbpasswd 时可以使用它。

应用实例如下。

添加 fred 的 Samba 账号。

```
# smbpasswd –a fred
```

9.5 SSH 服务器管理命令

9.5.1 sshd：OpenSSH 守护进程

作用：sshd 是 OpenSSH 守护进程，用于在不可信网络上提供安全的连接通道。

> **说明**
>
> sshd 守护进程通常由 root 用户启动，它监听来自客户端的连接，然后为每个连接派生一个子进程。子进程负责处理密钥交换、加密、认证、执行命令、数据交换等具体事务。sshd 的行为可以通过使用命令行选项和配置文件（默认是 sshd_config）进行控制，但命令行选项会覆盖配置文件中的设置。sshd 会在收到 SIGHUP 信号后重新读取配置文件，但是最初启动的命令行选项仍然有效（仍会覆盖配置文件中的设置）。

当某个用户成功登录后，sshd 将做一系列动作。如果在一个 tty 登录，并且没有指定执行任何命令，那么将打印上次登录时间和/etc/motd 的内容。/etc/motd 的内容可以通过配置文件或者～/.hushlogin 禁止。过程如下。

① 如果在一个 tty 登录，那么记录登录时间。

② 如果存在/etc/nologin 文件，并且登录用户不是 root，那么打印/etc/nologin 文件的内容并结束。

③ 切换身份至非特权用户运行。

④ 设置基本环境变量，如果配置文件允许用户修改环境变量，则处理～/.ssh/environment 文件并设置相应的环境变量（细节参见 sshd_config 手册页中的 PermitUserEnvironment 指令）。

⑤ 切换到该用户的家目录。

⑥ 如果存在～/.ssh/rc 文件则运行它，否则运行/etc/ssh/sshrc（如果存在）。

⑦ 如果上面两个文件都不存在则运行 xauth 程序。

⑧ 启动用户的 Shell 或运行指定的命令。

应用实例如下。

RHEL 5.0 提供了 SSH 服务器的 RPM 软件包，如果使用这种方法安装 SSH 服务器配置相关配置文件后，启动、停止、重新启动命令依次如下。

启动 sshd：

```
# service sshd start
启动 sshd：                               [确定]
```

关闭 sshd：

```
# service sshd stop：
停止 sshd：                               [确定]
```

重新启动 sshd：

```
停止 sshd：                               [确定]
启动 sshd：                               [确定]
```

如果希望 sshd 服务器随系统自动启动，可以使用如下命令：

```
#chkconfig sshd on
```

9.5.2　ssh-keygen：生成、管理和转换认证密钥

作用：ssh-keygen 用于为 SSH 生成、管理和转换认证密钥，包括 RSA 和 DSA 两种密钥。

用法：ssh-keygen [选项]

主要选项如下。

- -a trials：在使用-T 对 DH-GEX 候选素数进行安全筛选时，需要执行的基本测试数量。
- -B：显示指定的公钥/私钥文件的 bubblebabble 摘要。
- -b bits：指定密钥长度。对于 RSA 密钥，最小要求 768 位，默认是 2048 位。DSA 密钥必须恰好是 1024 位（FIPS186-2 标准的要求）。
- -C comment：提供一个新注释。
- -c：要求修改私钥和公钥文件中的注释。本选项只支持 RSA1 密钥。程序将提示输入私钥文件名、密语（如果存在）、新注释。

- -D reader：下载存储在智能卡 reader 里的 RSA 公钥。
- -e：读取 OpenSSH 的私钥或公钥文件，并以 RFC4716SSH 公钥文件格式在 stdout 上显示出来。该选项能够为多种商业版本的 SSH 输出密钥。
- -F hostname：在 known_hosts 文件中搜索指定的 hostname，并列出所有的匹配项。这个选项主要用于查找散列过的主机名/IP 地址，还可以和-H 选项联用打印找到的公钥的散列值。
- -f filename：指定密钥文件名。
- -G output_file：为 DH-GEX 产生候选素数。这些素数必须在使用之前使用-T 选项进行安全筛选。
- -g：在使用-r 打印指纹资源记录的时候使用通用的 DNS 用法。
- -H：对 known_hosts 文件进行散列计算。这将把文件中的所有主机名/IP 地址替换为相应的散列值。原来文件的内容将会添加一个“.old”扩展名后保存。这些散列值只能被 SSH 和 sshd 使用。这个选项不会修改已经经过散列的主机名/IP 地址，因此可以在部分公钥已经散列过的文件上安全使用。
- -i：读取未加密的 SSH-2 兼容的私钥/公钥文件，然后在 stdout 显示 OpenSSH 兼容的私钥/公钥。该选项主要用于从多种商业版本的 SSH 中导入密钥。
- -l：显示公钥文件的指纹数据。它也支持 RSA1 的私钥。对于 RSA 和 DSA 密钥，将会寻找对应的公钥文件，然后显示其指纹数据。
- -M memory：指定在生成 DH-GEXS 候选素数的时候最大内存用量（MB）。
- -N new_passphrase：提供一个新的密语。
- -P passphrase：提供（旧）密语。
- -p：要求改变某私钥文件的密语而不重建私钥。程序将提示输入私钥文件名、原来的密语，并要求两次输入新密语。
- -q：安静模式，用于在/etc/rc 中创建新密钥时。
- -R hostname：从 known_hosts 文件中删除所有属于 hostname 的密钥。这个选项主要用于删除经过散列的主机（参见-H 选项）的密钥。
- -r hostname：打印名为 hostname 的公钥文件的 SSHFP 指纹资源记录。
- -S start：指定在生成 DH-GEX 候选模数时的起始点（十六进制）。
- -T output_file：测试 Diffie-Hellmangroupexchange 候选素数（由-G 选项生成）的安全性。
- -t type：指定要创建的密钥类型。可以使用"rsa1"(SSH-1)"rsa"(SSH-2)"dsa"(SSH-2)。
- -U reader：把现存的 RSA 私钥上传到智能卡 reader。
- -v：详细模式。ssh-keygen 将会输出处理过程的详细调试信息。常用于调试模数的产生过程。重复使用多个-v 选项将会增加信息的详细程度（最大 3 次）。
- -W generator：指定在为 DH-GEX 测试候选模数时想要使用的 generator。
- -y：读取 OpenSSH 专有格式的公钥文件，并将 OpenSSH 公钥显示在 stdout 上。

应用实例如下。

1）生成 RSA 钥匙对

使用下列步骤来为 SSH 协议的版本 2 生成 RSA 钥匙对。从 OpenSSH 2.9 开始，它已成为

默认设置。

① 要生成 RSA 钥匙对与 SSH 协议的版本 2 合作，在 Shell 提示下输入下列命令：

```
ssh-keygen -t rsa
```

② 接受～/.ssh/id_rsa 的默认位置，输入一个与自己的账号密码不同的密码句，并再输入一次来确认。

③ 公钥被写入～/.ssh/id_rsa.pub，密钥被写入～/.ssh/id_rsa，绝不能把密钥出示给任何人。

④ 使用以下命令改变.ssh 目录的许可权限。

```
chmod 755 ~/.ssh
```

⑤ 把～/.ssh/id_rsa.pub 的内容复制到想连接的机器上的～/.ssh/authorized_keys 文件中。如果～/.ssh/authorized_keys 不存在，可以把～/.ssh/id_rsa.pub 文件复制到那个机器上的～/.ssh/authorized_keys 文件中。

⑥ 使用以下命令改变 authorized_keys 文件的许可权限：

```
chmod 644 ~/.ssh/authorized_keys
```

2）生成 DSA 钥匙对

使用下面的步骤来为 SSH 协议的版本 2 生成 DSA 钥匙对。

① 要生成用于 SSH 协议的版本 2 的 DSA 钥匙对，在 Shell 提示下输入下面的命令：

```
ssh-keygen -t dsa
```

② 接受～/.ssh/id_dsa 的默认位置。输入一个与自己的账号密码不同的密码句，并再输入一次来确认。

说明

密码句是用来验证用户的一串词汇和字符。密码句和一般密码的不同之处在于在密码句中可以使用空格或制表符。密码句通常比一般密码长，因为它们通常使用短语而不仅仅用一个词。

③ 公钥被写入～/.ssh/id_dsa.pub，密钥被写入～/.ssh/id_dsa，绝不能把密钥出示给任何人，这一点很重要。

④ 使用以下命令改变.ssh 目录的许可权限。

```
chmod 755 ~/.ssh
```

⑤ 把～/.ssh/id_dsa.pub 的内容复制到想连接的机器中的～/.ssh/authorized_keys 文件中。如果文件～/.ssh/authorized_keys 不存在，可以把～/.ssh/id_dsa.pub 文件复制到那个机器上的～/.ssh/authorized_keys 文件中。

⑥ 使用以下命令改变 authorized_keys 文件的许可权限。

```
chmod 644 ~/.ssh/authorized_keys
```

9.5.3 ssh：SSH 命令行登录工具

作用：ssh（SSH 客户端）用于登录远程主机，并且在远程主机上执行命令。它的目的是替换 rlogin 和 rsh，同时在不安全的网络之上、两个互不信任的主机之间，提供加密的、安全的通信连接。X11 连接和任意 TCP/IP 端口均可以通过此安全通道转发（forward）。

用法：ssh [-1246AaCfghkMNnqsTtVvXxY] [-b bind_address] [-c cipher_spec]
[-D port] [-e escape_char] [-F configfile] [-i identity_file]
[-L port:host:hostport] [-l login_name] [-m mac_spec] [-o option]
[-p port] [-R port:host:hostport] [-S ctl] [user@]hostname [command]

主要选项如下。

- -a：利用这个参数，可以使转发的认证代理无效。其结果是使已装入内存的密码无效。如果愿意，也可以针对每台主机指定这个参数，而不是使其成为全程的设置。
- -c cipher：可以通过这个参数为用户的网上传输指定一个用来加密数据的密钥对。
- -C：该参数将使 ssh 压缩所有通过 Secure Shell 客户端发送的数据，包括输入、输出、错误消息及转发数据。它使用 gzip 算法，压缩级别可通过设置配置文件中的参数 Compressicn Level 来指定。这对于缓慢的传输线路特别有用，但对于传输速度已经很快的网络则显得没有必要。同样，可以利用配置文件针对每台主机配置这个参数。
- -f：该参数将 ssh 连接送入后台执行。这在验证已经完成且 TCP/IP 转发已经建立的情况下会生效。这对在远程主机上启动 X 程序显得十分重要。其后用户将被提示要求输入密码（提供的认证代理不运行），然后将连接送往后台。
- -g：该参数允许远程主机通过端口转发与主机端口相连，通常情况下仅允许本地主机这样做。
- -k：这个参数使 Kerberos 转发功能无效。
- -l login_name：这个参数指定在远程主机上的登录用户名。默认情况下该用户名与用户在本地机上的用户名相同。它可以通过配置文件为不同的主机量体裁衣。这个参数是一个很实用的参数，因为许多人在不同的主机上有着不同的用户名。
- -L port:host:hostport：这个参数转发在本地主机上的指定端口与远程主机上的指定端口连接完成的信息。
- -n：这个参数与-f 参数类似。然而，在需要输入密码时它将不会工作。标准输入由/dev/null 重定向而得到，当 Secure Shell 客户端被送往后台时必须使用它。这个参数在用来发送 X 流量至远程主机时被普遍使用。最好在使用该参数时启用认证代理。
- -o option：在命令行参数未被定义时，该参数用来从配置文件中传递参数，包括 StrictHostke yChecking 和 Use Rsh，它们没有自己的命令行参数。
- -p port：可以指定哪个端口用作 Secure Shell 客户端与服务器的连接。默认值为 22，它为 Secure Shell 而保留。记住，除非特殊指定，用于服务器的端口号在/etc/services 文件中定义。可以通过配置文件针对不同的主机分别设置。
- -P：使用大于 1023 的端口号（非特权的端口）。不能使用这些端口进行 rhost 认证（无论是它自身还是与 RSA 的组合）。

– -q：这个参数选用哑模式。这意味着任何信息，包括警告与诊断信息，都不会被显示出来。在碰到认证或连接问题时，最好关闭该模式。
– -R port:host:hostport：该参数转发远程主机指定端口到本地主机指定端口上的连接。它和-L参数的工作过程刚好相反。套接字监听远程主机上的端口，只要针对该端口有一个连接，它将被转发到本地主机和主机端口。该转发端口可以在不同的主机上通过配置文件来进行不同的配置。特权端口只能在以超级账号登录到远程主机时转发。
– -t：该参数通过指定一个伪终端迫使 Secure Shell 客户端以交互模式工作，即使在给定命令的情况下也是如此。它被用于执行在远程主机上的基于屏幕的程序。
– -V：该参数打印出客户端的版本号后退出。
– -v：详尽模式。该参数使 ssh 打印出调试信息，在调试连接、认证及配置问题时该参数特别有用。
– -x：该参数用于 X 流量转发。X 被公认为在启动时是不安全的，对安全敏感的站点最好设置这个参数。该参数可以通过配置文件为每一台主机分别设置。

应用实例如下。

（1）SSH 首次登录过程。

```
ssh -l [your accountname on the remote host] [address of the remote host]
ssh －l cao@192.168.0.3
```

如果 OpenSSH 工作正常，用户会看到下面的提示信息：

```
The authenticity of host [hostname] can't be established.
Key fingerprint is 1024 5f:a0:0b:65:d3:82:df:ab:44:62:6d:98:9c:fe:e9:52.
Are you sure you want to continue connecting (yes/no)?
```

OpenSSH 告诉用户它不知道这台主机，但是不用担心这个问题，因为用户是第一次登录这台主机。输入“yes”，将把这台主机的“识别标记”添加到“～/.ssh/know_hosts”文件中，第二次访问这台主机的时候就不会再显示这条提示信息了。然后，SSH 提示输入远程主机上账号的密码。输入完密码之后，就建立了 SSH 连接，这之后就可以像使用 Telnet 那样使用 SSH 了。登录过程如图 9-7 所示。

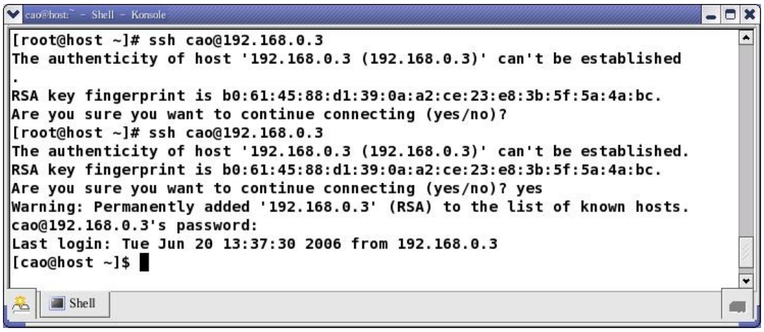


图 9-7　首次登录界面

（2）快速自动登录。

通常网络管理员要快速登录 SSH 服务器，可以像下面这样设定。如 www.xxx.com 域上有一个 cjh 用户，修改客户端配置文件/et/ssh/ssh_config，添加以下内容：

```
Host xxx
HostName www.xxx.com
User cjh
ForwardAgent yes
Compression yes
```

存盘退出。以后使用命令 ssh xxx，SSH 服务器会自动从配置文件中查询服务器名称，并且使用 cjh 的用户名称登录。

9.5.4 sftp：安全互动 FTP

作用：sftp 工具可以用来打开一次安全互动的 FTP 会话。它与 FTP 相似，只不过，它使用安全、加密的连接。它的一般语法是 sftp username@hostname.com。一旦通过验证，用户就可以使用一组和使用 FTP 相似的命令。

用法：sftp [选项]

主要选项如下。

- -c cipher：这是和 SSH 中定义的选项相同的选项，因为它直接被传送到 SSH。
- -d debug_level_spec：该选项定义用户接收的调试信息的数量，和 SSH 2 使用的选项相同。
- -p port：可以指定客户端连接到哪个端口上的服务器。默认设置为 22 号端口，该端口是为 Secure Shell 保留的。记住，除非另外指定，一般情况下用于服务器的该端口定义在文件/etc/services 中。它可以在配置文件中为每台主机单独指定端口。
- - v：冗余模式。与其他 Secure Shell 客户端一样，它使得 sftp 打印关于它的过程的调试信息。它打印出调试级为 2 的信息，这在调试连接、认证和配置问题时很有帮助，和用在 scp 中的相同选项类似。

应用实例如下。

使用 root 用户 sftp 登录主机 192.168.40.135。

```
sftp root@192.168.40.135
Connecting to 192.168.40.135...
root@192.168.40.135's password:
sftp> ?
Available commands:
cd path                         Change remote directory to 'path'
lcd path                        Change local directory to 'path'
chgrp grp path                  Change group of file 'path' to 'grp'
chmod mode path                 Change permissions of file 'path' to 'mode'
```

```
chown own path                  Change owner of file 'path' to 'own'
help                            Display this help text
get remote-path [local-path]    Download file
lls [ls-options [path]]         Display local directory listing
ln oldpath newpath              Symlink remote file
lmkdir path                     Create local directory
lpwd                            Print local working directory
ls [path]                       Display remote directory listing
lumask umask                    Set local umask to 'umask'
mkdir path                      Create remote directory
progress                        Toggle display of progress meter
put local-path [remote-path]    Upload file
pwd                             Display remote working directory
exit                            Quit sftp
quit                            Quit sftp
rename oldpath newpath          Rename remote file
rmdir path                      Remove remote directory
rm path                         Delete remote file
symlink oldpath newpath         Symlink remote file
version                         Show SFTP version
!command                        Execute 'command' in local shell
!                               Escape to local shell
?                               Synonym for help
```

上面显示 sftp 子命令。

sftp 相关子命令的使用方法如表 9-1 所示。

表 9-1　sftp 相关子命令的使用方法

子命令名称	说　明
cd	变换目录
mkdir	建立子目录
ls	显示文件名称
pwd	列出当前目录名称
rm	删除文件
ln	建立文件链接
charp	修改文件组属性
chmod	修改文件权限
rename	修改文件或目录名称
lpwd	显示当前位置
lmkdir	建立本地目录

续表

子命令名称	说　　明
put	上传文件
get	下载文件
exit	离开远程服务器

9.5.5　scp：将文件复制到远程主机或本地主机

作用：scp 可以将文件复制到远程主机或本地主机。所以，这取决于用户所要发送的文件的位置（要么在远程主机上，要么在本地），必须指定用户名、主机名、目录和文件。听上去这有点复杂，并且需要记住许多东西。但只要正确地使用了这些参数，就会得到正确的结果。比如说笔者会经常使用 scp 在家中的 Linux 笔记本与 ISP 中心上的服务器的账号之间进行文件复制。

用法：scp [选项]文件 1[...]文件 2

主要选项如下。

- -a：为每个文件打开显示功能。信息将显示复制每个文件的进程。如果用户希望全程地打开这个功能，可以使用-Q 选项。
- -A：该选项关闭针对每个文件的设计信息显示功能。如果在程序脚本中使用 scp，可以使用-q 选项完全关闭这个功能。
- -B：批处理模式。该模式防止对密码的输入而使你能在程序脚本中使用 scp。
- -c cipher：这是一个与 ssh 命令相同的选项，因为它被直接传递给 ssh。
- -C：这也是一个与 ssh 命令相同的选项，因为它被直接传递给 ssh。
- -iidentity_file：这个选项也和在 ssh 中被定义的同名选项一样，其选项值将被直接传递到 ssh。
- -L：该选项使用大于 1023 的端口，即那些非特权端口。用户不能将它用于 rhosts 认证。它对那些通过无配置文件的防火墙连接是十分有帮助的。该选项与 ssh 的-p 选项一致。也可以在配置文件中写上 Use PriledgePort=no 或在命令行中加上“-o UsePriviledgePort=no”来实现该选项的功能。
- -ossh_option：该选项的指定值将被直接传递给 ssh。
- -p：保存源主机上的文件属性，属性包括修改时间、读/写时间及初始文件的模式。该选项直接从 rcp 而来，在用户希望备份文件且需要确保文件完整性时它是十分有效的。
- -P port：可以指定客户端在哪个端口上与服务器相连接。默认值为端口 22，它为 Secure Shell 所保留。除非特殊指定，会在文件/etc/services 中为服务器指定该端口。也可以为每台主机分别配置，但与 ssh 稍有不同，这是因为 rcp 使用-p 选项来保存文件的属性。
- -q：该选项关闭统计信息显示功能，它并不针对单个文件，而是为所有的文件服务。
- -Q：该选项打开统计信息显示功能。与-q 类似，针对单个文件打开该功能时，使用-a 选项。
- -r：搜索完整的目录并复制该目录下的所有文件。

– -S path_to_ssh：该选项指定能找到 ssh 的路径。在将 ssh 安装到非标准目录下时该选项十分有用。
– -v：详尽模式。该模式使 ssh 和 scp 显示进程的调度信息。在调试连接、认证和配置所出现的问题时，该选项很有用处。

应用实例如下。

将文件 options.cipcb 从服务器复制到客户机主机的/etc/cipe/目录中。

```
# scp root@192.168.1.9:/etc/cipe/options.cipcb0 /etc/cipe/options.cipcb0
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.9' (RSA) to the list of known hosts.
root@192.168.1.9's password:
options.cipcb0 100% |*******************************|61 00:00
```

9.6 squid：代理服务器 squid 守护进程

作用：squid 是代理服务器 squid 守护进程。

用法：squid [-hvzCDFNRYX] [-d level] [-s | -l facility] [-f config-file] [-u port] [-k signal]

主要选项如下。

– -a port：指定新的 http_port 值。该选项覆盖了来自 squid.conf 的值。然而请注意，你能在 quid.conf 里指定多个值？-a 选项仅仅覆盖配置文件里的第一个值（该选项使用字母 a 是因为在 Harvest cache 里 HTTP 端口被叫作 ASCII 端口）。
– -d level：让 squid 将它的调试信息写到标准错误（假如配置了，就是 cache.log 和 syslog）。level 参数指定了显示在标准错误里的消息的最大等级。在多数情况下，d1 工作良好。
– -f file：指定另一个配置文件。
– -h：显示用法。
– -k function：指示 squid 执行不同的管理功能。功能参数是 reconfigure、rotate、shutdown、interrupt、kill、debug、check 和 parse 中之一。
– + reconfigure：导致运行中的 squid 重新读取配置文件。
– + rotate：导致 squid 滚动它的日志，包括关闭日志、重命名和再次打开日志。
– + shutdown：发送关闭 squid 进程的信号。
– + interrupt：立刻关闭 squid，不必等待活动会话完成。
– + kill：发送 KILL 信号给 squid，这是关闭 squid 的最后保证。
– + debug：将 squid 设置成完全的调试模式，假如你的 cache 很忙，它能迅速地用完你的磁盘空间。
– + check：简单地检查运行中的 squid 进程，返回的值显示 squid 是否正在运行。
– +parse：简单地解析 squid.conf 文件，如果配置文件包含错误，进程将返回非零值。
– -s：激活日志记录到 syslog 进程。squid 使用 LOCAL4 syslog 设备。0 级别调试信息以优

先级 LOG_WARNING 被记录，1 级别消息以 LOG_NOTICE 被记录，更高级的调试信息不会被发送到 syslogd。用户可以在/etc/syslogd.conf 文件里使用 local4.warning /var/log/squid.log 接口。

- -u port：指定另一个 ICP 端口号，覆盖 squid.conf 文件里的 icp_port。
- -v：打印版本信息。
- -z：初始化 cache，或者交换目录。在首次运行 squid 或者增加新的 cache 目录时，必须使用该选项。
- -C：阻止安装某些信号句柄，它们捕获特定的致命信号，如 SIGBUS 和 SIGSEGV。正常情况下，这些信号被 squid 捕获，以便它能干净地关闭。
- -D：禁止初始化 DNS 测试。正常情况下，squid 直到验证它的 DNS 可用才能启动。该选项阻止了这样的检测。也可以在 squid.conf 文件里改变或删除 dns_testnames 选项。
- -F：让 squid 拒绝所有的请求，直到它重新建立起存储元数据。假如用户的系统很忙，该选项可以减短重建存储元数据的时间。然而，如果用户的 cache 很大，重建过程可能会花费很长的时间。
- -N：阻止 squid 变成后台服务进程。
- -R：阻止 squid 在绑定 HTTP 端口之前使用 SO_REUSEADDR 选项。
- -V：激活虚拟主机加速模式。类似于 squid.conf 文件里的 httpd_accel_host virtual 指令。
- -X：强迫完整调试模式，如在 squid.conf 文件里指定 debug_options ALL,9 一样。
- -Y：在重建存储元数据时，返回 ICP_MISS_NOFETCH 代替 ICP_MISS。忙碌的父 cache 在重建时，该选项可以导致最少的负载。

应用实例如下。

配置并保存好 squid.conf 后，可以用以下命令启动 squid：

```
/etc/rc.d/init.d/squid start
```

同样，也可以用下列脚本停止运行 squid 或重新启动 squid：

```
/etc/rc.d/init.d/squid stop
/etc/rc.d/init.d/squid restart
```

9.7 DHCP 服务器管理命令

9.7.1 dhcpd：DHCP 服务器守护进程

作用：dhcpd 是 DHCP 服务器守护进程。

用法：dhcpd [选项]

主要选项如下。

- -p port：指定 DHCP 运行端口。默认端口号是 67，服务器在比指定的 UDP 端口大一位的端口号码上把回应传输给 DHCP 客户。譬如，如果用户接受了默认的端口 67，那么

服务器在端口 67 上监听请求，然后在端口 68 上回应客户。如果用户在此处指定了一个端口号码来使用 DHCP 转发代理，用户所指定的 DHCP 转发代理的监听端口也必须是同一端口。
- -f：把守护进程作为前台进程运行。这在调试时最常用。
- -d：把 DHCP 服务器守护进程记录到标准错误描述器中。这在调试时最常用。如果它没有指定，日志将被写入/var/log/messages。
- -cf config-file：指定配置文件的位置。默认位置是/etc/dhcpd.conf。
- -lf lease-file：指定租期数据库文件的位置。如果租期数据库文件已存在，在 DHCP 服务器每次启动时使用同一个文件至关重要。强烈建议只在无关紧要的机器上为调试目的才使用该选项。默认的位置是/var/lib/dhcp/dhcpd.leases。
- -q：在启动该守护进程时，不要显示整篇版权信息。

应用实例如下。

（1）启动 DHCP 服务。

```
#/sbin/service dhcpd start
```

（2）停止 DHCP 服务器。

```
#/sbin/service dhcpd stop
```

（3）让守护进程在引导时自动启动。

```
#chkconfig named on
```

9.7.2　dhclient：DHCPv6 客户端守护进程

作用：dhclient 守护进程通过设置 IP 地址来实现 DHCP 客户机，并通过使用 DHCP 协议来实现其他参数。

用法：dhclient [选项] interface

主要选项如下。
- -c ConfigurationFile：指定要用的配置文件。
- -r <all | ipv6 addresses>：允许 DHCP 守护进程释放所有的 IPv6 地址。
- -d：显示 dhclient 调试信息。
- -D：显示 dhclient 调试详细信息。

应用实例如下。

（1）DHCP 获取 IP 命令。

```
#dhclient eth0
```

（2）释放 IP 地址。

```
$ dhclient -r
```

9.7.3　dhcp6c：DHCPv6 客户端守护进程

作用：dhcp6c 守护进程通过设置 IPv6 地址来实现 DHCP 客户机，并通过使用 DHCP 协议来实现其他参数。

用法：dhcp6c [选项] interface

主要选项如下。

- -c ConfigurationFile：指定要用的配置文件。
- -r <all | ipv6 addresses>：允许 dhcp6c 守护进程释放所有的 IPv6 地址。
- -d：显示 dhcp6c 调试信息。
- -D：显示 dhcp6c 调试详细信息。

应用实例如下。

启用 IPv6 客户端。

```
#dhcp6c eth0
```

9.8　FTP 服务器管理命令

9.8.1　FTP 服务器端管理命令

Linux 下的 FTP 服务器软件包括 vsftpd、ProFTPD 和 wu-ftpd。目前这 3 种 Linux 服务器端的 FTP 软件在因特网中的占有率超过了 95%以上，最流行的是 vsftpd。

1．vsftpd：vsftpd 守护进程

作用：vsftpd 命令是 vsftpd 守护进程。

用法：vsftpd [configuration file]

主要选项如下。

configuration file：设置配置文件，默认是/etc/vsftpd/vsftpd.conf。

应用实例如下。

（1）启动 FTP 服务器。

```
# service vsftpd start
启动 vsftpd：                                      [确定]
```

（2）关闭 FTP 服务器。

```
# service vsftpd stop
关闭 vsftpd：                                      [确定]
```

（3）重新启动 FTP 服务器。

```
# service vsftpd restart
```

```
关闭 vsftpd：                                            [确定]
为 vsftpd 启动 vsftpd：                                  [确定]
```

2．ProFTPD：服务器管理命令

作用：ProFTPD 全称为 Professional FTP Daemon，是针对 wu-ftpd 的弱项而开发的，除了改进安全性，还具备许多 wu-ftpd 没有的特点。

用法：proftpd

应用实例如下。

（1）关闭服务器。

```
#/etc/rc.d/init.d/proftpd stop
```

（2）启动服务器。

```
# /etc/rc.d/init.d/proftpd start
```

（3）重新加载修改后的配置文件 etc/proftpd.conf 并且启动服务器。

```
#/etc/rc.d/init.d/ proftpd reload
```

3．wu-ftpd：wu-ftpd 控制守护进程

作用：wu-ftpd 命令是 wu-ftpd 控制守护进程。

用法：wu-ftpd [选项]

主要选项如下。

- -d：当 FTP 服务器出错时，将错误信息写入系统的 syslog 文件中。
- -l：将每次 FTP 客户端进行连接的信息写入系统的 syslog 文件中。
- -t：设置 FTP 客户端连接几分钟无操作就切断连接。
- -a：使 Wu-FTP 使用/etc/ftpaccess 的设定。
- -A：使 Wu-FTP 不使用/etc/ftpaccess 的设定。
- -L：将 FTP 客户端连线后所执行的程序记录在系统的 syslog 文件中。
- -I：将 FTP 客户端上载文件的日志记录在/usr/adm.xferlog 文件中。
- -o：将 FTP 客户端下载文件的日志记录在/usr/adm/xferlog 文件中。

应用实例如下。

使 xinetd 监控 wu-ftpd 服务器的运行。

```
#chkconfig wu-ftpd on
```

这个命令会修改/etc/xinetd.d/wu-ftpd 中的 disable 为 no，并动态让 xinetd 读入。

启动 wu-ftpd 的方法如下：

```
#ftpshut now        关闭 FTP 服务器，当有用户连到 FTP 时必须先关闭 FTP 服务器
#service xinetd restart
#ftprestart         重启 FTP 服务器
```

9.8.2 FTP 客户端和 FTP 监控命令

1．FTP：FTP 客户端工具在本地主机和远程主机之间传送文件

作用：FTP 命令使用“文件传送协议（FTP）”在本地和远程主机或远程主机之间传送文件。FTP 使用 ARPANet 的标准文件传输协议，该网络就是现在 Internet 的前身。

用法：ftp [-v] [-d] [-i] [-n] [-g] [-k realm] [-f] [-x] [-u] [-t]] [主机名称或 IP 地址]

主要选项如下。

- -g：禁用文件名中的元字符拓展。解释元字符可参考为扩展（有时叫作文件名匹配替换）文件名。
- -i：关闭多文件传送中的交互式提示。
- -n：防止在起始连接中的自动登录。
- -q：允许用户指定 send_file 子例程必须用于在网络上发送文件。只有当文件在无保护的情况下以二进制方式发送时此标志才适用。
- -v：显示远程服务器的全部响应，并提供数据传输的统计信息。当 FTP 命令的输出是到终端（如控制台或显示）时，此显示方式是默认方式。
- -f：导致转发凭证。如果 Kerberos 5 不是当前认证方法，则此标志将被忽略。
- -d：将有关 FTP 命令操作的调试信息发送给 syslogd 守护进程。如果指定-d 标志，必须编辑/etc/syslog.conf 文件并添加 user.info FileNameOR 中的一项。
- -k realm：如果远程站的域不同于本地系统的域，系统将允许用户指定远程站的域。因此，域和 DCE 单元是同义的。如果 Kerberos 5 不是当前认证方法，则此标志将被忽略。

FTP 子命令如下。

- !<local command>：在本地系统运行命令。
- $<macro>：运行宏。
- ascii：将文件传输模式设为 ASCII。
- bell：文件传输完成后响铃。
- binary：将文件传输模式设为二进制方式。
- case：打开 case 功能。
- dir：打印路径名。
- glob：在文件传输时使用通配符。
- hash：每传输 1024 字节打印一个“#”。
- help：打印帮助文件。
- lcd<directory>：改变本地系统路径。
- ls：打印远程主机目录。
- lpwd：打印本地主机工作路径。
- prompt：交互式提示。
- verbose：激活冗余模式。
- delete <filename>：在远程计算机中删除文件。
- get <filename>：将文件从远程系统下载到本地系统。

– account [password]：应用其他用户名/密码以获取额外的对远程系统的访问权限。
– append localfile [remotefile]：将本地文件添加到远程文件中（文件在远程计算机中）。
– close, disconnect：关闭远程连接。
– cd <directory>：改变远程系统目录名。
– cd up：进入父目录。
– debug <level>：设置调试级别。
– macdef <macroname>：定义宏。
– mdel <filename (s)>：删除多个宏文件。
– mdir <filename (s)>：打印多个文件的路径。
– mget <filename (s)>：将多个文件从远程系统下载到本地系统。
– mkdir <directory>：在远程计算机上创建路径。
– mput <filename (s)>：将多个文件上载到远程计算机。
– open <site>：打开到<site>的连接。
– put <filename>：将文件上载到远程计算机。
– pwd：显示当前工作路径。
– user [user] [password]：登录到远程计算机系统。

通过对上述命令的运用，用户可以使用 FTP 完成多种功能。

应用实例如下。

要调用 FTP 命令，请登录系统，打印工作目录，更改工作目录，将文件传送类型设定为 ASCII 向远程主机发送本地文件，将工作目录改为父目录，然后退出，请输入：

```
$ ftp canopus
连接 canopus.austin.century.com.
220 canopus.austin.century.com FTP 服务器(4.1 版 1991 年 11 月 23 日周六 12:52:09 CST）就绪。
名称（canopus:eric）：dee
331 dee 所需要的密码。
密码：
230 用户 dee 登录。
ftp> pwd
257 "/home/dee"为当前目录。
ftp> cd 桌面
250 CWD 命令顺利完成。
ftp> 类型 ascii
200 类型设置为 A。
ftp>发送 typescript
200 PORT 命令顺利完成。
150 为 typescript（128.114.4.99,1412）打开数据连接。
226 传输完成。
```

```
ftp> cdup
250 CWD 命令顺利完成。
ftp> bye
221 再见。$
```

2．tfpd：传输文件

作用：tftp 是简单的文字模式 FTP 程序，它所使用的指令和 FTP 类似。

用法：tftp [选项] [主机名称或 IP 地址]

主要选项如下。

- connect host [port]：为传输设置 host（以及可选的 port）。请注意，与 FTP 协议不同，TFTP 协议不保持传输间的连接。因此，connect 命令实际上不会创建连接，仅仅记忆将使用哪一台主机进行传输。不一定非要使用 connect 命令，可将远程主机指定为 get 或 put 命令的一部分。
- mode transfer-mode：设置传输模式。transfer-mode 可以是 ASCII 或 Binary（默认为 ASCII）。
- -g tftp：命令会在覆盖现有本地文件之前提示确认。

应用实例如下。

（1）从远程主机复制一个文本文件并将它写到本地文件。

```
#tftp -g newsched host1 /home/john/schedule
```

在此例中，从远程主机 host1 复制/home/john/schedule 文件并写到本地文件 newsched 中。

（2）从远程主机复制文件并将输出重定向到本地主机的标准输出。

```
tftp -g - host3 /etc/hosts
```

如果复制成功，将在屏幕上显示类似于以下的信息：

```
192.100.13.3 nameserver
192.100.13.3 host2
192.100.13.5 host1
192.100.13.7 host3
192.100.13.3 timeserver
Received 128 bytes in 0.4 seconds
```

在此例中，从远程主机 host3 复制/etc/hosts 文件并将输出重定向到本地主机的标准输出。

（3）从远程主机复制文件、将文件输送给 grep 命令并将它写到本地文件。

```
utftp -g - host1 /home/john/schedule | grep Jones > jones.todo
```

在此例中，从远程主机 host1 复制/home/john/schedule 文件，然后文件被输送给 grep 命令，并写入到本地文件 jones.todo 中。

（4）从其他系统复制文件。

```
tftp -p /home/jeanne/test host2 /tmp/test
```

如果复制成功，在屏幕上将显示类似于以下的信息：

```
Sent 94146 bytes in 6.7 seconds
```

在此例中，/home/jeanne/test 文件被发送到远程主机 host2 的/tmp 目录中。

（5）将一个二进制文件复制到另一个系统。

```
tftp -p core host3 /tmp/core image
```

如果复制成功，在屏幕上将显示类似于以下的信息：

```
Sent 309295 bytes in 15 seconds
```

在此例中，从当前目录将二进制文件 core 发送到远程主机 host3 的/tmp 目录中。

3．ftpshut：在指定的时间关闭 FTP 服务器

作用：在指定的时间关闭 FTP 服务器。

用法：ftpshut [-l min] [-d min] time [warning-message...]

主要选项如下。

– -l min：在 FTP 关闭服务之前的几分钟内，尝试建立新的 FTP 连接均不被接受。
– -d min：在 FTP 关闭服务之前的几分钟内，已经建立的 FTP 连接将被中止。
– time：在多少时间后，服务器将关闭 ftp 服务，格式有以下两种。
  Ø ＋number：经过 number 分钟后关闭。
  Ø MMHH：在今天 MM:HH 服务器将关闭。

说明

ftpshut 提供系统管理者在设置的时间关闭 FTP 服务器，且能在关闭之前发出警告信息通知用户。关闭时间若设置后为“none”，则会立即关闭服务器。如果采用“+30”的方式来设置，则表示服务器在 30 分钟之后关闭。依此类推，假设使用“1130”的格式，则代表服务器会在每日的 11：30 关闭，时间格式为 24 小时制。FTP 服务器关闭后，在/etc 目录下会产生一个名为 shutmsg 的文件，把它删除后即可再度启动 FTP 服务器的功能。

应用实例如下。

过 30 分钟后，FTP 服务将关闭，在这之前的 20 分钟不可接受任何新的 FTP 连接，已经建立的 FTP 连接在服务关闭前 10 分钟强制断线，并在客户端显示“FTP Server Will shutdown at time”。

```
#ftpshut -l 20 -d 10 +30 "FTP Server Will shutdown at time"
```

4．ftpcount：显示 FTP 登录用户数量

作用：执行这项指令可得知目前用 FTP 登录系统的人数及 FTP 登录人数的上限。

应用实例如下。

显示 FTP 登录用户数。

```
#ftpcount
11
```

5．ftpwho：显示 FTP 登录用户信息

作用：执行这项指令可得知目前用 FTP 登录系统的用户信息。

应用实例如下。

显示 FTP 登录用户信息。

```
#ftpwho
Goodcjh
```

9.9 E-mail 服务器管理命令

9.9.1 sendmail：为本地或网络交付传送邮件

作用：sendmail 命令接收格式化文本消息并把它们发送到一个或更多的用户。在网络上使用时，sendmail 命令转换消息的头信息用法以匹配目的系统的要求。程序使用地址的语法和内容来确定目的系统的网络。

用法：sendmail [mode] [flags]address

主要选项如下。

- mode：从下面“模式”介绍中描述的模式中选择的模式，仅可以指定一种模式。默认值是-bm。
- flags：从下面“标志”介绍中描述的标志中选择的标志，可以指定几个标志。
- address：收件人地址，可以指定多个地址。

1．mode（模式）

sendmail 在下列模式之一中运行。默认值是-bm，表示以常规方式传递邮件。

- -ba：进入 ARPANET 模式。所有输入行必须以 CR-LF 结尾，并且生成的所有邮件在末尾都具有 CR-LF。同时将检查“From:”和“Sender:”字段，查找发件人的姓名。
- -bd：作为守护程序运行。sendmail 将在后台进行派生和运行，在套接字 25 上监听传入的 SMTP 连接。
- -bD：作为守护程序运行，但是在前台运行。
- -bh：输出永久主机状态数据库。
- -bH：清除永久主机状态数据库。
- -bi：初始化邮件别名文件的别名数据库。newaliases 与 sendmail -bi 等同。
- -bm：以常规方式传递邮件（默认方式）。
- -bp：输出邮件队列列表。mailq 与 sendmail -bp 等同。

- -bs：标准输入和输出使用 RFC821 中所述的 SMTP 协议。此标志暗含了与 SMTP 兼容的 ba 标志的所有操作。
- -bt：在地址测试模式下运行。此模式将读取地址并且显示分析时的步骤，它用于调试配置表。
- -bv：仅验证名称，即不尝试收集或传递邮件。验证模式通常用于验证用户或邮件列表。

2．flags（标志）

sendmail 采用下列标志。

- -Btype：设置正文类型。type 可以是 7BIT 或 8BITMIME。
- -Cfile：使用备用配置文件。如果指定了备用配置文件，sendmail 将拒绝以超级用户身份运行。
- -dX：将调试值设置为 X。X 的形式也可以是 category.level（例如，-d56.12）。较低的 level 或 category 生成较少输出，而较高的 level 或 category 生成较多输出。category 的默认值为 0，level 的默认值为 1。
- -Ffullname：设置发件人的全名。
- -fname：将“from”人（即邮件的发件人）的姓名设置为 name。如果-f 选项的用户不是“受信任”用户（通常是 root、daemon 和 network），并且使用-f 选项设置的 name 和实际发送邮件的人的登录名不同，它将导致邮件标头中出现 X-Authentication-Warning。
- -G：在不进行任何处理的情况下，转发邮件。
- -hN：将跃点计数设置为 *N*。每次处理邮件时，跃点计数将递增。当其达到一定限制时，邮件将返回并且带有错误消息，这是别名循环造成的。
- -i：忽略接收的邮件中只包含点的各行。如果正在从文件进行读取，应该设置此标志。
- -Ltag：指定在 syslog 消息中使用的标识符。此标识符将设置为 tag。
- -n：不设置别名。
- -Ndsn：设置传递状态通知条件。可以将 dsn 设置为下列有效状态。
  - Ø never：表示没有通知。
  - Ø failure：如果传递失败。
  - Ø delay：如果传递延迟。
  - Ø success：当邮件成功传递时。
- -Ooption=value：将配置选项 option 设置为指定的 value。
- -ox=value：将选项 x 设置为指定的 value（选项将在下面的“处理选项”中介绍）。
- -p protocol：设置用于接收邮件的协议的名称。
- -qtime：以指定的间隔处理队列中已保存的邮件。如果忽略 time，将处理队列一次。time 指定为带标记的数字，s 表示秒，m 表示分钟，h 表示小时，d 表示天，w 表示星期。
- -Rreturn：设置在邮件退回的情况下要返回的邮件信息量。
- -t：读取收件人的邮件。将扫描行 To:、Cc:和 Bcc:，查找收件人地址。
- -U：初始（用户）提交。从用户代理（如 mail 或 elm）调用 sendmail 时，应该始终设置此标志。从网络传递代理（如 rmail）调用 sendmail 时，永远不应设置此标志。
- -v：进入详细信息模式，将公布别名扩展等。

– --：停止处理命令标志并使用其余参数作为地址。

应用实例如下。

（1）sendmail 服务器启动命令。

```
# /etc/rc.d/init.d/sendmail start
```

（2）sendmail 服务器重新启动命令。

```
# /etc/rc.d/init.d/sendmail restart
```

（3）sendmail 服务器停止命令。

```
# /etc/rc.d/init.d/sendmail stop
```

（4）显示待寄邮件的清单。

```
# sendmail –bp
```

9.9.2 mail：E-mail 管理程序

作用：mail 是一个文字模式的邮件管理程序，操作的界面不像 elm 或 pine 那么容易使用，但功能尚称完整。mail 具有 ed 的命令语法回忆功能，只是用消息替换了行。

用法：mail [-iInv] [-s subject] [-c cc-addr] [-b bcc-addr] to-addr... mail [-iInNv] -f [name] mail [-iInNv] [-u user]

主要选项如下。

– -v：详尽模式。传输邮件的细节都呈现在用户的终端上。
– -i：忽略 tty 中断信号。这对于在嘈杂的电话线路上使用 mail 特别有用。
– -I：强迫 mail 以交互模式运行，即使其不是通过终端输入的。特别地，正在发送邮件时，“～”特殊字符只在交互模式下才起作用。
– -n：禁止在启动时读取/etc/mail.rc。
– -N：当阅读邮件或编辑邮件夹时禁止消息头的初始化显示。
– -s：在命令行上指定主题（仅把-s 标识后的第一个参数用作主题，注意对包含空格的主题加上引号）。
– -c：发送复件（carboncopy）给用户 list。
– -b：发送隐藏的复写副本（blindcarboncopy）给用户 list，list 应为以逗号分隔的名字列表。
– -f：读入 mbox（或指定文件）中的内容进行处理；当 quit 时，mail 会把未删除的消息写回该文件。
– -u：相当于 mail -f /var/spool/mail/user。

应用实例如下。

（1）启动 mail 程序并列出邮箱中的消息。

```
#mail
```

mail 命令列出在系统邮箱中的每条消息。邮件系统然后显示邮箱提示符（？）来表示等待输入。当看到此提示符时，请输入邮箱子命令。要查看子命令列表，请输入：

```
?
```

本条目列出#mail 子命令。

（2）将消息 letter 发送到收件人 user1@host1，并抄送给 user2@host2 和 user3@host3。

```
#mail -c "user2@host2 user3@host3" user1@host1<letter
```

（3）查看个人邮箱的内容。

```
#mail –f
```

此命令显示在个人邮箱$HOME/mbox 中的消息列表。

（4）查看指定邮件文件夹的内容。

```
#mail -f +dept
```

此命令显示在 dept 文件夹中的消息列表。

（5）将消息发送到本地系统的用户。

```
#mail ron
```

完成将消息输入到用户 ron 时，按 Enter 键及“.”（点号）或 Ctrl+D 组合键来退出编辑器并发送消息。要确定某用户是否在本地系统中，请在/etc/passwd 文件中查看该用户名。如果消息发送成功，将不会收到通知。如果消息无法发送，将发送一个错误信息。

（6）将一个文件通过邮件发送到本地系统的另一位用户。

```
#mail karen < letter1
```

此命令将文件 letter1 的内容发送到本地系统的用户 karen。发送文件后，#mail 程序将显示命令行提示符。

（7）将消息发送到一个远程系统用户。

```
#mail dale@zeus
```

现在可以创建一个到 dale 的消息。在本例中，发送消息到远程系统 zeus 上的 dale 用户。要将消息发送到通过网络连接到你的系统的另一个系统上的用户，你必须知道那个人的登录标识和另一个系统的名称。

9.9.3　mailq：显示待寄邮件的清单

作用：mailq 可列出待寄邮件的清单，包括邮件 ID、邮件大小、邮件保存时间、寄信人、收信人，以及邮件无法寄出的原因，提供管理员可参考的信息。

用法：mailq [-v]

主要选项如下。

-v：打印详细信息。此处添加消息优先级和一个单字符指示符（+或空白），指示符用来表明是否在消息的第一行发送一个警告消息。除此以外，也许混合了多余的行，其中包含显示控制用户的信息。此处显示的用户拥有基于这个消息执行的，以及自这个命令扩展的别名的名称

的任何程序。

mailq 命令显示邮件队列中的消息列表。每个消息打印出来的第一行如下。

– 可能带有一个状态字符的消息，用于此主机上的内部标识符。
– 以字节为单位的消息的大小。
– 将消息接收到队列中的日期和时间。
– 消息的信包发送方。

第二行显示使消息滞留在队列中的错误消息。如果消息是第一次显示，那么将不会显示错误消息。状态字符是以下之一。

– *：表明正在处理作业。
– X：表明负载过高，不能处理作业。
– -：表明作业过长，不能处理作业。

9.9.4 mailstats：显示关于邮件流量的统计信息

作用：mailstats 命令显示当前邮件的统计信息。统计信息开始显示时根据 ctime 指定的格式读取信息。

用法：mailstats [-o] [-p] [-C CfFile] [-f StatFile]

主要选项如下。

– -C cfFile：指定使用 cfFile 代替默认的 sendmailcf 文件。
– -f StatFile：指定使用 StatFile 代替 sendmail.cf 文件中指定的统计信息文件。
– -o：指定不在输出中显示的邮件程序的名称。
– -p：在程序可读方式下输出信息且清除统计信息。

应用实例如下。

根据 ctime 指定的格式显示当前邮件的统计信息。

```
# mailstats
Statistics from Sun Oct 21 13:32:47 2007
 M   msgsfr  bytes_from   msgsto    bytes_to  msgsrej msgsdis msgsqur Mailer
 9      29         30K      29         30K       0       0       0  local
=====================================================================
 T      29         30K      29         30K       0       0       0
 C       1                  0                    0
```

命令输出时，邮件程序的统计信息显示一行，包含以下字段。

– M：包含邮件程序数。
– msgsfr：包含本地机器从指明的邮件程序接收到的消息数。
– bytes_from：包含本地机器从指明的邮件程序接收到的消息中的千字节数。
– msgsto：包含从本地机器使用指明的邮件程序发送的消息数。
– bytes_to：包含从本地机器使用指明的邮件程序发送的消息中的千字节数。
– msgsrej：包含被拒绝的消息数。

– msgsdis：包含被废弃的消息数。
– Mailer：包含邮件程序的名称。

9.9.5　mutt：电子邮件管理程序

作用：mutt 是一个文字模式的电子邮件管理程序，提供了全屏幕的操作界面。
用法：mutt[选项]
主要选项如下。

– -a<文件>：在邮件中加上附加文件。
– -b<地址>：指定密件副本的收信人地址。
– -c<地址>：指定副本的收信人地址。
– -f<邮件文件>：指定要载入的邮件文件。
– -F<配置文件>：指定 mutt 程序的设置文件，而不读取预设的.muttrc 文件。
– -h：显示帮助。
– -H<邮件草稿>：将指定的邮件草稿送出。
– -i<文件>：将指定文件插入邮件内文中。
– -m<类型>：指定预设的邮件信箱类型。
– -n：不要去读取程序培植文件（/etc/Muttrc）。
– -p：在 mutt 中编辑完邮件后，若不想将邮件立即送出，可将该邮件暂缓寄出。
– -R：以只读的方式开启邮件文件。
– -s<主题>：指定邮件的主题。
– -v：显示 mutt 的版本信息及当初编译此文件时所给予的参数。
– -x：模拟 mailx 的编辑方式。
– -z：与-f 参数一并使用时，若邮件文件中没有邮件则不启动 mutt。

应用实例如下。
使用交换界面的 mutt 如图 9-8 所示。

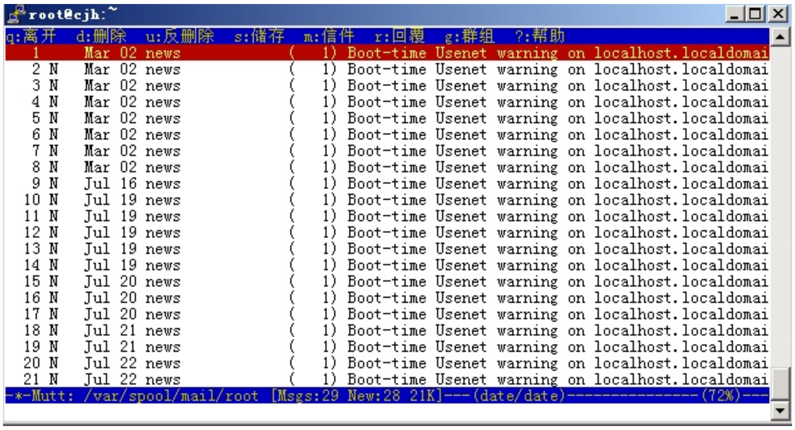


图 9-8　mutt 界面

mutt 控制键如下。

q：离开；d：删除；u：反删除；s：存储；m：信件；r：回复；g：群组；?：帮助。

9.10 Linux 防火墙管理工具 iptables 和 firewalld

9.10.1 iptables：Linux 防火墙管理工具

作用：iptalbes 用来设置、维护和检查 Linux 内核的 IP 包过滤规则。

用法：iptables [-t table] CMD [match] [-j target]

iptables 命令比较复杂，也比较重要，我们把它分解进行介绍。

1．-t table

t 参数用来指定规则表，内建的规则表有 3 个，分别是 Nat、Mangle 和 Filter，当未指定规则表时，则一律视为 filter。各个规则表的功能如表 9-2 所示。

表 9-2　各规则表的功能

表　名	表功能描述	链　名	功能描述
Filter	包过滤	FORWARD	转发数据包
		INPUT	处理到防火墙的数据包
		OUTPUT	处理防火墙发出的数据包
Nat	主要用处是网络地址转换，包括 IP NAT 和 IP 伪装	PREROUTING	PREROUTING 主要用在这样一种情况下：你有一个合法的 IP 地址，要把对防火墙的访问重定向到其他的机器上（如 DMZ）。也就是说，我们改变的是目的地址，以使包能重路由到某台主机
		POSTROUTING	POSTROUTING 的作用和 MASQUERADE 完全一样，只是计算机的负荷稍微多一点。因为对每个匹配的包，MASQUERADE 都要查找可用的 IP 地址，而不像 SNAT 用的 IP 地址是配置好的
		OUTPUT	进行网络地址转换用于防火墙产生的数据包，不常使用
Mangle	TCP 头修改	PREROUTING POSTROUTING OUTPUT INPUT FORWARD	改变不同的包及包头的内容，如 TTL、TOS 或 MARK。注意 MARK 并没有真正地改动数据包，它只是在内核空间为包设了一个标记。防火墙内的其他规则或程序（如 tc）可以使用这种标记对包进行过滤或高级路由。不常使用

2．CMD

CMD 是 iptables 命令的最重要部分。它告诉 iptables 命令要做什么，例如，插入规则、将规则添加到链的末尾或删除规则。在使用 iptables 时，如果必需的参数没有输入就按 Enter 键，那么它就会给出一些提示信息，告诉你需要哪些参数。如表 9-3 所示是命令选项及其作用。

表 9-3　命令选项及其作用

命令名称	功能说明
-A, --append	在所选择的链末添加规则。当源地址或目的地址是以名字而不是 IP 地址的形式出现时，若这些名字可以被解析为多个地址，则这条规则会和所有可用的地址结合
-D, --delete	从所选链中删除规则。有两种方法指定要删除的规则：一是把规则完完整整地写出来；二是指定规则在所选链中的序号（每条链的规则都各自从 1 开始被编号）
-R, --replace	在所选中的链里指定的行上（每条链的规则都各自从 1 开始被编号）替换规则。它主要的用处是试验不同的规则。当源地址或目的地址是以名字而不是 IP 地址的形式出现时，若这些名字可以被解析为多个地址，则这条命令会失败
-I, --insert	根据给出的规则序号向所选链中插入规则。如果序号为 1，规则会被插入链的头部，其实默认序号就是 1
-L, --list	显示所选链的所有规则。如果没有指定链，则显示指定表中所有的链。如果什么都没有指定，就显示默认表中所有的链。精确输出受其他参数影响，如-n 和-v 等
-F, --flush	清空所选的链。如果没有指定链，则清空指定表中的所有链。如果什么都没有指定，就清空默认表所有的链
-Z, --zero	把指定链（如未指定，则认为是所有链）的所有计数器归零
-N, --new-chain	根据用户指定的名字建立新的链
-X, --delete-chain	删除指定的用户自定义链，这个链必须没有被引用，如果被引用，在删除之前必须删除或者替换与之有关的规则。如果没有给出参数，这条命令将会删除默认表所有非内建的链
-P, --policy	为链设置默认的 target（可用的是 DROP 和 ACCEPT），这个 target 称为策略。所有不符合规则的包都被强制使用这个策略。只有内建的链才可以使用规则。但内建的链和用户自定义的链都不能被作为策略使用
-E, --rename-chain	对自定义的链进行重命名，原来的名字在前，新名字在后。这仅仅是改变链的名字，对整个表的结构、工作没有任何影响
-h	显示帮助信息
-v, --verbose	这个选项使输出详细化，常与--list 连用。与--list 连用时，输出中包括网络接口的地址、规则的选项、TOS 掩码、字节和包计数器，其中计数器是以 KB、MB、GB 为单位的。如果-v 和--ppend、--insert、--delete 或--replace 连用，iptables 会输出详细的信息，告诉用户规则是如何被解释的、是否正确地插入等
x, --exact	使--list 输出中的计数器显示准确的数值，注意此选项只能和--list 连用
-n, --numeric	使输出中的 IP 地址和端口以数值的形式显示，而不是默认的名字，如主机名、网络名、程序名等。注意此选项也只能和--list 连用
--line-numbers	只能和--list 连用的选项，其作用是显示出每条规则在相应链中的序号，这样用户就可以知道序号了
-c, --set-counters	在创建或更改规则时设置计数器
--modprobe	此选项告诉 iptables 探测并装载要使用的模块。这是一个非常有用的选项，如果 modprobe 命令不在搜索路径中，就要用到了。有了这个选项，在装载模块时，即使有一个需要用到的模块没装载上，iptables 也知道要去搜索

3．匹配（match）

iptables 命令的可选 match 部分指定信息包与规则匹配所应具有的特征（如源和目的地址、协议等）。可把它们归为 5 类：第一类是 generic matches（通用的匹配），适用于所有的规则；第二类是 TCP matches，顾名思义，这只能用于 TCP 包；第三类是 UDP matches，当然它只能用在 UDP 包上；第四类是 ICMP matches，是针对 ICMP 包的；第五类比较特殊，针对的是状态，指所有者和访问的频率限制等。在此只介绍通用匹配，熟悉了通用匹配，其他的几种匹配也就比较容易理解了。如表 9-4 所示是常用匹配和功能。

表 9-4　常用匹配和功能

参　数	说　明
-p, --protocol	匹配指定的协议。默认设置为 ALL，相应数值是 0，但要注意这只代表匹配 TCP、UDP、ICMP
-d, --dst, --destination	以 IP 源地址匹配包
-i, --in-interface	以包进入本地所使用的网络接口来匹配包。要注意这个匹配操作只能用于 INPUT、FORWARD 和 PREROUTING 这 3 个链
-o, --out-interface	以包离开本地所使用的网络接口来匹配包
--sport, --source-port	用来比对封包的来源埠号，可以比对单一埠，或一个范围，例如，--sport 22:80，表示从 22 到 80 埠之间都算符合条件。如果要比对不连续的多个埠，则必须使用--multiport 参数
--dport, --destination-port	用来比对封包的目的端口号
--tcp-flags	比对 TCP 封包的状态旗号，参数分为两部分，第一部分列举出想比对的旗号，第二部分则列举前述旗号中哪些有被设定，未被列举的旗号必须是空的。TCP 状态旗号包括 SYN（同步）、ACK（应答）、FIN（结束）、RST（重设）、URG（紧急）、PSH（强迫推送）等，均可使用于参数中，除此之外还可以使用关键词 ALL 和 NONE 进行比对
--icmp-type	用来比对 ICMP 的类型编号，可以使用代码或数字编号来进行比对
-m limit --limit	用来比对某段时间内封包的平均流量（单位时间为秒）
-m state --state	用来比对联机状态。联机状态共有 4 种：INVALID、ESTABLISHED、NEW 和 RELATED。 INVALID 表示该封包的联机编号无法辨识或编号不正确。 ESTABLISHED 表示该封包属于某个已经建立的联机。 NEW 表示该封包想要起始一个联机（重设联机或将联机重导向）。 RELATED 表示该封包是属于某个已经建立的联机，所建立的新联机（如 FTP-DATA 联机）必定源自某个 FTP 联机
-m owner --gid-owner	用来比对来自本机的封包，是否为某特定使用者群组所产生的

4．目标（target）

我们已经知道，目标是由规则指定的操作，那些与规则匹配的信息包执行这些操作。除了允许用户定义的目标之外，还有许多可用的目标选项，用于建立高级规则的目标，如 LOG、REDIRECT、MARK、MIRROR 和 MASQUERADE 等。如表 9-5 所示是常用目标及其说明。

表 9-5　常用目标及其说明

目　标	说　明
ACCEPT	当信息包与具有 ACCEPT 目标的规则完全匹配时，会被接受（允许它前往目的地），并且它将停止遍历链（虽然该信息包可能遍历另一个表中的其他链，并且有可能在那里被丢弃）。该目标被指定为-j ACCEPT
DROP	当信息包与具有 DROP 目标的规则完全匹配时，会阻塞该信息包，并且不对它做进一步处理。该目标被指定为-j DROP
REJECT	该目标的工作方式与 DROP 目标相同，但它比 DROP 好。和 DROP 不同的是，REJECT 不会在服务器和客户机上留下死套接口。另外，REJECT 将错误消息发回给信息包的发送方。该目标被指定为-j REJECT
RETURN	在规则中设置的 RETURN 目标让与该规则匹配的信息包停止遍历包含该规则的链。如果链是如 INPUT 之类的主链，则使用该链的默认策略处理信息包。它被指定为-jump RETURN
MASQUERADE	改写封包来源 IP 为防火墙网卡 IP
LOG	将封包相关信息记录在/var/log 中
SNAT	改写封包来源 IP 为某特定 IP 或 IP 范围
DNAT	这个目标只适用于 nat 表的 POSTROUTING 链。它规定修改包的源地址（此连接以后所有的包都会被影响），停止对规则的检查
MIRROR	镜射封包，也就是将来源 IP 与目的地 IP 对调后，将封包送回。进行完此处理动作后，将会中断过滤程序
QUEUE	中断过滤程序，将封包放入队列，交给其他程序处理。透过自行开发的处理程序，可以进行其他应用
RETURN	结束在目前规则链中的过滤程序，返回主规则链继续过滤。如果把自定义规则链看成一个子程序，那么这个动作就相当于提早结束子程序并返回到主程序中
MARK	将封包标上某个代号，以便提供作为后续过滤的条件判断依据。进行完此处理动作后，将会继续比对其他规则
TOS	用来设置 IP 包的首部八位 tos。只能用于 mangle 表

应用实例如下。

1）激活#iptables 服务

防火墙规则只有在#iptables 服务运行的时候才能被激活。要手工启动服务，使用以下命令：

```
#service #iptables restart
```

要确保它在系统引导时启动，使用以下命令：

```
#chkconfig --level 345 #iptables on
```

ipchains 服务不能和#iptables 服务同时运行。要确定 ipchains 服务被禁用，执行以下命令：

```
#chkconfig --level 345 ipchains off
```

2）查看规则集

要查看系统中现有的#iptables 规则集，可以运行以下命令：

```
# #iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source destination

Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
```

上面是没有定义规则时#iptables 的样子。

3）允许到防火墙上的端口 80 的通信

```
#iptables -A INPUT -p tcp -m tcp --sport 80 -j ACCEPT
#iptables -A OUTPUT -p tcp -m tcp --dport 80 -j ACCEPT
```

这会允许用户浏览通过端口 80 通信的网站。要允许到安全网站（如 https://www.example.com/）的访问，还必须打开端口 443。

```
#iptables -A INPUT -p tcp -m tcp --sport 443 -j ACCEPT
#iptables -A OUTPUT -p tcp -m tcp --dport 443 -j ACCEPT
```

4）配置 NAT 规则

多数机构从它们的 ISP 处得到数量有限的可公开选路的 IP 地址。iptables 提供了选路发送和转发策略，用户可以实施它们来防止对网络资源的变相利用。FORWARD 策略允许管理员控制分组可以被选路发送到 LAN 内的哪些地方。例如，要允许整个 LAN 的转发（假定防火墙/网关在 eth1 上有一个内部 IP 地址），可以设置以下规则：

```
#iptables -A FORWARD -i eth1 -j ACCEPT
#iptables -A FORWARD -o eth1 -j ACCEPT
```

按照默认设置，Red Hat Linux 内核中的 IPv4 策略禁用了对 IP 转发的支持，这会防止运行 Red Hat Linux 的机器成为专用边缘路由器。要启用 IP 转发，请运行以下命令：

```
#sysctl -w net.ipv4.ip_forward=1
```

如果该命令是通过 Shell 提示运行的，那么其设置在重新引导后就不会被保存。可以通过编辑/etc/sysctl.conf 文件来永久性地设置转发。寻找并编辑以下行，把 0 改成 1：

```
net.ipv4.ip_forward = 0
```

执行以下命令来启用 sysctl.conf 文件中的改变：

```
#sysctl -p /etc/sysctl.conf
```

这会允许 LAN 节点彼此通信。不过，它们没有被允许和外界（如互联网）通信。要允许带有专用 IP 地址的 LAN 节点和外部的公共网络通信，配置防火墙的 IP 伪装（IP Masquerading），这会把来自 LAN 节点的请求都伪装成防火墙的外部设备（在这个例子中是 eth0）的 IP 地址。

```
#iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

5）DMZ 和#iptables 配置

还可以设置一些把交通选路发送到某些机器（如专用 HTTP 或 FTP 服务器）的规则，这些机器最好是位于停火区域（De-Militarized Zone，DMZ）的和内部网络分开的机器。要设置一条把所有进入的 HTTP 请求都选路发送到 IP 地址为 10.0.4.2、端口为 80（LAN 192.168.1.0/24 范围之外）的专用 HTTP 服务器的规则，网络地址转换（NAT）会调用 PREROUTING 表来把这些分组转发到恰当的目的地。

```
#iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT –to
-destination 10.0.4.2:80
```

使用这项命令，所有来自 LAN 以外的到端口 80 的 HTTP 连接都会被选路发送到和内部网络分离的另一个网络上的 HTTP 服务器上。这种网络分段会比允许到内部网络中的机器上的 HTTP 连接更安全。如果 HTTP 服务器被配置接受安全连接，那么端口 443 也必须被转发。

6）病毒和假冒 IP 地址

可以更精心地设计一些规则来控制到 LAN 内指定子网的访问，甚至到指定机器的访问。还可以限制某些类似特洛伊木马、蠕虫及其他客户/服务器病毒的可疑服务联系它们的服务器。例如，有些特洛伊木马会扫描端口 31 337～31 340（即黑客语言中的 elite 端口）上的服务。合法服务都不使用这些非标准端口来进行通信，阻塞这些端口能够有效地减少网络上可能被感染的机器和它们的远程主服务器进行独立通信的机会。

```
#iptables -A OUTPUT -o eth0 -p tcp --dport 31337 --sport 31337 -j DROP
#iptables -A FORWARD -o eth0 -p tcp --dport 31337 --sport 31337 -j DROP
```

还可以阻塞试图假冒用户所在 LAN 的专用 IP 地址混入的连接。例如，如果用户的 LAN 使用 192.168.1.0/24 范围，面向互联网的网络设备（如 eth0）上就可以设置一条规则来放弃到那个设备的使用用户所在 LAN 的 IP 范围的分组。因为默认策略是拒绝转发分组的，所有到面向外界的设备（eth0）的假冒 IP 地址都会被自动拒绝。

```
#iptables -A FORWARD -s 192.168.1.0/24 -i eth0 -j DROP
```

9.10.2　ip6tables：IPv6 版本的 iptables 工具

作用：ip6tables 命令是 IPv6 版本的 iptables 工具。

用法：ip6tables [-t table] CMD [match] [-j target]

> **说明**
>
> ip6tables 命令的用法与 iptables 命令相同。其语法在各方面都和 IP Tables 相同，只不过 IPTables 支持 128 位的地址。

应用实例如下。

（1）启动 IP6 Tables。

使用 IP6 Tables 的第一步是启动 IP6Tables 服务，可以使用以下命令：

```
#service ip6tables start
```

必须关闭 IP Tables 服务，才能专门使用 IP6Tables 服务。

```
#service iptables stop
#chkconfig iptables off
```

要使 IP6 Tables 在系统引导时默认启动，使用 chkconfig 来改变服务的运行级别状态。

```
#chkconfig --level 345 ip6tables on
```

（2）启用在识别 IPv6 的网络服务器上的 SSH 连接。

```
#ip6tables -A INPUT -i eth0 -p tcp -s 3ffe:ffff:100::1/128 --dport 22 -j
ACCEPT
```

9.10.3 iptables-save：iptables 列表存储

作用：iptables-save 用来将 IP Table 转存为可以简单解析的格式，输出到标准输出 STDOUT。可以使用 Shell 的 I/O 重定向功能来写入文件。

用法：iptables-save [-c] [-t table]

主要选项如下。

– -c, --counters：在输出中包含所有报文和字节计数的当前值。
– -t, --table tablename：限制只输出一个表。如果不指定，会输出所有可能的表。

应用实例如下。

把规则集保存到/etc/iptables-save 中，而且还包括计数器。

```
# iptables-save -c > /etc/iptables-save
```

9.10.4 iptables-restore：装载由 iptables-save 保存的规则集

作用：iptables-restore 用来装载由 iptables-save 保存的规则集。它只能从标准输入接收输入，而不能从文件接收。

用法：iptables-restore [-c] [-t table]

主要选项如下。

– -n, --noflush：不刷新表中从前的内容。
– -c, --counters：恢复所有报文和字节计数的值。

应用实例如下。

将规则集正确地装入内核。

```
# iptables-restore
```

9.10.5　firewall-cmd：下一代防火墙管理工具

作用：firewall-cmd 是下一代防火墙管理工具。

用法：firewall-cmd[选项]

主要选项如下。

– -h, --help：打印帮助信息。

– -V, --version：打印版本信息。

– -q, --quiet：退出。

应用实例如下。

（1）显示服务列表。

Amanda、Ftp、Samba 和 TFTP 等最重要的服务已被 Firewalld 提供相应的服务，可以使用如下命令查看：

```
# firewall-cmd --get-services
cluster-suite pop3s bacula-client smtp ipp radius bacula ftp mdns samba dhcpv6-client https openvpn imaps samba-client http dns telnet libvirt ssh ipsec ipp-client amanda-client tftp-client nfs tftp libvirt-tls
```

（2）允许 SSH 服务通过。

```
# firewall-cmd --enable service=ssh
```

（3）禁止 SSH 服务通过。

```
# firewall-cmd --disable service=ssh
```

（4）临时允许 Samba 服务通过 600 秒。

```
# firewall-cmd --enable service=samba --timeout=600
```

（5）打开一个 TCP 的 8080 端口。

```
#firewall-cmd --enable ports=8080/tcp
```

（6）显示当前服务。

```
#firewall-cmd --list-services
dhcpv6-client ssh
```

（7）添加 HTTP 服务到内部区域（internal）。

```
# firewall-cmd --permanent --zone=internal --add-service=http
# firewall-cmd –reload
```

（8）显示支持的区域列表。

```
# firewall-cmd --get-zones
block drop work internal external home dmz public trusted
```

（9）设置为家庭区域。

```
# firewall-cmd --set-default-zone=home
```

（10）查看当前的区域。

```
#firewall-cmd --get-active-zones
```

（11）设置当前区域的接口。

```
#firewall-cmd --get-zone-of-interface=enp03s
```

（12）显示所有公共区域（public）。

```
# firewall-cmd --zone=public --list-all
```

（13）临时修改网络接口 enp03s 为内部区域（internal）。

```
# firewall-cmd --zone=internal --change-interface=enp03s
```

（14）永久修改网络接口 enp03s 为内部区域（internal）。

```
# firewall-cmd --permanent --zone=internal --change-interface=enp03s
```

（15）打开端口。

打开 443/tcp 端口在内部区域（internal）。

```
# firewall-cmd --zone=internal --add-port=443/tcp
# firewall-cmd –reload
```

（16）端口转发。

```
# firewall-cmd --zone=external --add-masquerade
# firewall-cmd --zone=external
--add-forward-port=port=22:proto=tcp:toport=3777
```

上述两个命令的含义是，首先启用伪装（masquerade），然后把外部区域（external）的 22 端口转发到 3777。

（17）关闭服务的方法。

用户也可以关闭目前还不熟悉的 Firewalld 防火墙，使用老的 IPTables，步骤如下：

```
# systemctl stop firewalld
# systemctl disable firewalld
# yum install iptables-services
# systemctl start iptables
# systemctl enable iptables
```

9.10.6　arptables：管理 ARP 包过滤的软件

作用：arptables 用于管理 ARP 包过滤的软件。

用法：arptables [-t table] -[AD] chain rule-specification [options]
　　　arptables [-t table] -I chain [rulenum] rule-specification [options]
　　　arptables [-t table] -R chain rulenum rule-specification [options]
　　　arptables [-t table] -D chain rulenum [options]
　　　arptables [-t table] -[LFZ] [chain] [options]
　　　arptables [-t table] -N chain
　　　arptables [-t table] -X [chain]
　　　arptables [-t table] -P chain target [options]
　　　arptables [-t table] -E old-chain-name new-chain-name

常用参数如下。

一类为命令类。

- -A, --append chain rule-specification：追加规则。
- -D, --delete chain rule-specification：删除指定规则。
- -D, --delete chain rulenum：删除指定位置的规则。
- -I, --insert chain [rulenum] rule-specification：插入规则。
- -R, --replace chain rulenum rule-specification：替换规则。
- -L, --list [chain]：列出规则。
- -F, --flush [chain]：删除所有规则。
- -Z, --zero [chain]：清空所有计数。
- -N, --new-chain chain：新建链。
- -X, --delete-chain [chain]：删除链。
- -P, --policy chain target：指定默认目标。
- -E, --rename-chain old-chain new-chain：重命名链。
- -h：帮助。

另一类为参数。

- -s, --source [!] address[/mask]：源地址。
- -d, --destination [!] address[/mask]：目的地址。
- -z, --source-hw [!] hwaddr[mask]：源 MAC。
- -y, --target-hw [!] hwaddr[mask]：目的 MAC。
- -i, --in-interface [!] name：收到这个包的网卡。
- -o, --out-interface [!] name：要发送这个包的网卡。
- -a, --arhln [!] value[mask]。
- -p, --arpop [!] value[mask]。
- -H, --arhrd [!] value[mask]。
- -w, --arpro [!] value[value]。
- -j, --jump target：跳到目标。
- -c, --set-counters: PKTS BYTES 计数。

说明

arptables 用于建立、获取、修改内核的 ARP 包处理表。有几个不同的表，每个表分别含有几条内建的处理链，同时允许用户自定义处理链。每条链是一些规则的列表，每条规则匹配特定的包。每条规则指定一个对匹配的包的操作。这个操作也叫作“目标”，这个目标也可以是跳转到同一个表中的另外的链。

内建目标：ACCEPT、DROP、QUEUE、RETURN 是几个最基本的目标。ACCEPT 指接受这个包；DORP 指丢掉这个包；QUEUE 指把包传到用户空间（如果内核指定了的话）；RETURN 指返回到上一条链，接着执行上一条链跳转过来的那条规则的下一条规则。每条链都有一个默认目标，当包经过所有规则都没被匹配时，则发给默认目标表：一边至少有一个内建的表（filter 表），-t 常常用于指定要操作的表。filter 表有两条内建的链：IN 和 OUT，IN 用于处理进入本机的数据包，OUT 用于处理本机发出去的包。

应用实例如下。

（1）将源是 aaaa 而目的不是 cccc 的转发包伪装为源地址是 bbbb。

```
# iptables -t nat -A POSTROUTING -s aaaa ! -d cccc -j SNAT --to bbbb
```

（2）将源不是 cccc 而目的是 aaaa 的包丢掉。

```
# arptables -A IN ! -s cccc -d aaaa -j DROP
```

（3）将源是 aaaa 而目的不是 cccc 的请求包发给 bbbb。

```
# arptables -A OUT -s aaaa ! -d cccc -j mangle --mangle-ip-s bbbb
```

9.11 SELinux 管理命令

9.11.1 setenforce：设置 SELinux 模式

作用：setenforce 命令用来设置 SELinux 模式。

用法：setenforce [Enforcing | Permissive | 1 | 0]

主要选项如下。

- Enforcing：设置 SELinux 模式为强制。
- Permissive：设置 SELinux 模式为允许。
- 1：设置 SELinux 模式为强制。
- 0：设置 SELinux 模式为允许。

应用实例如下。

（1）设置 SELinux 为允许模式，临时关闭 SELinux。

```
#setenforce 0
```

（2）设置 SELinux 为强制模式。

```
#setenforce 1
```

9.11.2　getenforce：查看 SELinux 模式

作用：getenforce 命令用来查看 SELinux 模式。

用法：getenforce

getenforce 命令无选项。

应用实例如下。

查看当前 SELinux 模式。

```
#getenforce
Enforcing
```

上述代码显示 SELinux 模式为强制。

9.11.3　setsebool：设置 SELinux 布尔值

作用：setsebool 命令用来设置 SELinux 布尔值。

用法：setsebool [-P] boolean value | bool1=val1 bool2=val2...

主要选项如下。

-P：设定该项目永久套用。

应用实例如下。

（1）设置布尔值变量 ftpd_disable_trans 为 on。

```
# setsebool ftpd_disable_trans=on(on 也可用 1)
```

检查一下。

```
#getsebool ftpd_disable_trans
ftpd_disable_trans --> on
```

（2）设置布尔值变量 ftpd_disable_trans 为 off。

```
#setsebool -P ftpd_disable_trans=off(off 也可用 0)
```

检查一下。

```
#getsebool ftpd_disable_trans
ftpd_disable_trans --> off
```

9.11.4　getsebool：查看 SELinux 布尔值

作用：getsebool 命令用来查看 SELinux 布尔值。

用法：getsebool [-a] [boolean]

主要选项如下。

- -a：显示所有布尔值。
- Boolean：指定要显示的布尔值名称。

应用实例如下。

（1）显示所有布尔值。

```
# getsebool -a
NetworkManager_disable_trans --> off
allow_console_login --> off
allow_cvs_read_shadow --> off
allow_daemons_dump_core --> on
allow_daemons_use_tty --> on
allow_domain_fd_use --> on
allow_execheap --> off
allow_execmem --> on
allow_execmod --> off
allow_execstack --> on
allow_ftpd_anon_write --> off
allow_ftpd_full_access --> off
allow_ftpd_use_cifs --> off
allow_ftpd_use_nfs --> off
…
```

（2）显示 allow_execmem 布尔值。

```
# getsebool allow_execmem
allow_execmem --> on
```

9.11.5 togglesebool：翻转 SELinux 布尔值

作用：togglesebool 命令用来翻转 SELinux 布尔值。

用法：togglesebool boolean

主要选项如下。

boolean：要转换的布尔值名称。

应用实例如下。

转换 allow_execmem 布尔值。

```
# togglesebool allow_execmem
allow_execmem: inactive
```

查看一下。

```
# getsebool allow_execmem
allow_execmem --> off
```

此时 allow_execmem 布尔值从 on 转换为 off。

9.11.6　sestatus：SELinux 状态查看工具

作用：sestatus 命令是 SELinux 状态查看工具。

用法：sestatus[选项]

主要选项如下。

- -v：检查/etc/sestatus.conf 文件和进程列表。
- -b：显示当前使用的布尔值状态。

应用实例如下。

（1）查询系统 SELinux 目前的状态。

```
# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   enforcing
Mode from config file:          permissive
Policy version:                 21
Policy from config file:        targeted
输出说明:
SELinux 状态:                   enabled
SELinuxfs 文件系统挂载点:        /selinux
当前模式:                        enforcing
配置文件模式:                    permissive
策略版本:                        21
配置文件中的策略:                targeted
```

（2）显示当前使用的布尔值状态。

```
# sestatus -v
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   enforcing
Mode from config file:          permissive
Policy version:                 21
Policy from config file:        targeted
Process contexts:
```

```
Current context:
root:system_r:unconfined_t:              SystemLow-SystemHigh
Init context:                            system_u:system_r:init_t
/sbin/mingetty                           system_u:system_r:getty_t
/usr/sbin/sshd
system_u:system_r:unconfined_t:          SystemLow-SystemHigh
File contexts:
Controlling term:                        root:object_r:devpts_t
/etc/passwd                              system_u:object_r:etc_t
/etc/shadow                              system_u:object_r:shadow_t
/bin/bash                                system_u:object_r:shell_exec_t
/bin/login                               system_u:object_r:login_exec_t
…
```

（3）显示当前所有布尔值状态。

```
# sestatus -b |more
SELinux status:                          enabled
SELinuxfs mount:                         /selinux
Current mode:                            permissive
Mode from config file:                   permissive
Policy version:                          21
Policy from config file:                 targeted
Policy booleans:
NetworkManager_disable_trans             off
allow_console_login                      off
allow_cvs_read_shadow                    off
…
```

9.11.7 avcstat：显示 AVC 统计信息

作用：avcstat 命令用来显示 AVC 统计信息。

用法：avcstat[-c] [-f status_file] [interval]

主要选项如下。

- -c：显示累计值。
- -f status_file：指定 AVC 文件位置，默认是/selinux/avc/cache_stats。
- Interva：间隔时间，以秒为单位。

说明

命令 avcstat 提供了从启动以来的访问向量缓存（Access Vector Cache，AVC）的统计信息，能指定时间间隔实时查看 AVC 的信息。AVC 信息存在于文件/selinux/avc/cache_stats 中，能使用命令 cat /selinux/avc/cache_stats 查看文件内容，或使用命令 avcstat 直接查看 AVC 信息。

应用实例如下。

使用命令 avcstat 查找当前和每隔 2 秒的 AVC 信息。

```
# avcstat 2
   lookups        hits          misses       allocs     reclaims   frees
   5637478        5625903        11575        11809       8912     11309
       362        362            0                0          0        0
```

9.11.8　audit2why：转换审计消息

作用：audit2why 命令转换成一个 SELinux 的审计信息描述为何访问被拒绝。

用法：audit2why [-p policy] < /var/log/audit/audit.log

主要选项如下。

-p policy：指定 SELinux 策略文件。

应用实例如下。

将登录文件的内容读进来分析，并输出分析的结果。

```
$ /usr/sbin/audit2why < /var/log/audit/audit.log
```

9.11.9　audit2allow：生成策略允许规则

作用：audit2allow 命令生成策略允许规则。

用法：audit2allow [选项]

主要选项如下。

- --version：查看版本信息。
- -h, --help：显示帮助信息。
- -a, --audit：从 audit 日志中读取输入，不能与-i 共享。
- -d, --dmesg：从 dmesg 中读取输入，不能与--audit 或--input 共享。
- -i INPUT：指定输入来源不能与-a 共享。
- -l：仅读取自上次重新加载 SELinux 以来的信息。
- -r,--requires：用于规则的一般要求。
- -m：指定为哪个模块生成规则，如 local、postfix、httpd、clamd、myaqld 等。

- -M MODULE_PACKAGE：生成一个模块的规则包，不能与-o 及-m 共享。
- -o OUTPUT：向某个已经存在的文件中添加规则，不能与-M 共享。
- -R,--reference：生成 refpolicy 格式的输出。
- -v：解释已生成的输出。
- -e,--explain：详尽解释已生成的输出。
- -t TYPE：只处理与指定类型一致的消息。
- --perm-map=PERM_MAP：指定 perm 映象的文件名。
- --interface-info=INTERFACE_INFO：界面信息的文件名。
- --debug：将生成的模块用于-M。

说明

audit2allow 命令是一个很重要的用 python 写的命令，主要用来处理日志，把日志中违反策略动作的记录转换成 access vector，对开发安全策略非常有用。在 refpolicy 里，它的功能比以前有了很大的扩展。

应用实例如下。

（1）使用 audit2allow 生成模块政策。

```
$ cat /var/log/audit/audit.log | audit2allow -m local > local.te
$ cat local.te
#cat local.te
module local 1.0：
require {
type clock_device_t：
type unlabeled_t：
type usr_t：
type ping_t：
type qemu_t：
type user_home_t：
class capability sys_resource：
class file { write ioctl read getattr }：
class chr_file { read ioctl }：
}
#============= ping_t ==============
allow ping_t usr_t:file { read write }：
#============= qemu_t ==============
allow qemu_t clock_device_t:chr_file { read ioctl }：
allow qemu_t self:capability sys_resource：
```

```
allow qemu_t unlabeled_t:file { read write ioctl };
allow qemu_t user_home_t:file { read write getattr };
```

（2）使用 audit2allow 产生和建立模块政策。

```
# cat /var/log/audit/audit.log | audit2allow -M local
******************** IMPORTANT ***********************
To make this policy package active, execute:
semodule -i local.pp
```

9.11.10　load_policy：装载策略

作用：load_policy 命令用来装载策略。

用法：load_policy [-bq]

主要选项如下。

- -b：重置策略的布尔值。
- -q：不显示报警信息。

应用实例如下。

设置布尔值变量 ftpd_disable_trans 为 off。

```
#setsebool -P ftpd_disable_trans=off(off 也可用 0)
```

检查一下。

```
#getsebool ftpd_disable_trans
ftpd_disable_trans --> off
```

装载策略到 Linux 内核。

```
# load_policy
```

9.11.11　semanage：管理 SELinux 策略

作用：semanage 命令把 Linux 用户名捆绑到 SELinux 用户。

用法：semanage {login|user|port|interface|fcontext|translation} -l [-n]

semanage login -{a|d|m} [-sr] login_name

semanage user -{a|d|m} [-LrRP] selinux_name

semanage port -{a|d|m} [-tr] [-p protocol] port | port_range

semanage interface -{a|d|m} [-tr] interface_spec

semanage fcontext -{a|d|m} [-frst] file_spec

semanage translation -{a|d|m} [-T] level

主要选项如下。

- -a, -add：添加一个 OBJECT 记录 NAME。
- -d, -delete：删除一个 OBJECT 记录 NAME。

- -m, -modify：修改一个 OBJECT 记录 NAME。
- -l, -list：显示一个 OBJECT 列表。
- -h, -help：显示帮助。
- -p, -proto：指定端口（TCP 或者 UDP）。
- -P, -prefix：定义家目录的标签。
- -L, -level：定义 SELinux 的默认级别。
- -R, -roles：SELinux 规则。
- -s, -seuser：SELinux 用户。
- -t, -type：SELinux 类型。

应用实例如下。

（1）设置让 SELinux 停止阻止一个守护进程监听和默认端口不同的端口。

这个例子使用 Apache 作为范例，并且假设在/etc/httpd/conf/httpd.conf 文件中已经配置好监听 81 端口（HTTP 默认监听 80 端口）。

①使用 semanage 命令搜索 SELinux 关于 Apache 的端口配置。

```
#semanage port -l | grep 80
```

②在端口列表中增加一个 SELinux 允许 Apache 通过的端口。

```
#semanage port -a -t http_port_t 81
```

这个操作对其他的守护进程同样生效，例如 Squid（假设在/etc/squid/squid.conf 文件中设置了监听其他端口，这里用 num 表示）：

```
#semanage port -l | grep 3128*
#semanage port -a -t http_cache_port_t num
```

（2）查看 SELinux 用户映射关系。

```
# semanage user -l
Labeling       MLS/            MLS/
SELinux    User  Prefix  MCS Level MCS Range    SELinux Roles
root       user  s0      SystemLow-SystemHigh  system_r sysadm_r user_r
system_u   user  s0      SystemLow-SystemHigh  system_r
user_u     user  s0      SystemLow-SystemHigh  system_r sysadm_r user_r
```

9.11.12 semodule：管理策略模块

作用：semodule 命令用来管理策略模块。

用法：semodule [选项]... MODE [MODES]...

主要选项如下。

- -R, --reload：强制重新装入策略。
- -B, --build：强制重新生成策略。
- -i,--install=MODULE_PKG：安装一个新模块。
- -u,--upgrade=MODULE_PKG：升级一个新模块。
- -r,--remove=MODULE_NAME：删除一个新模块。
- -l,--list-modules：显示所有模块。

说明

这是一个功能强大的策略管理工具，有了它，即使没有策略的源代码，也是可以管理安全策略的。

应用实例如下。

显示所有模块。

```
# semodule -l
amavis          1.1.0
ccs             1.0.0
clamav          1.1.0
iscsid          1.0.0
postgrey        1.1.0
prelude         1.0.0
pyzor           1.1.0
qemu            1.1.2
razor           1.1.0
ricci           1.0.0
smartmon        1.1.0
spamassassin    1.9.0
virt            1.0.0
zosremote       1.0.0
```

9.11.13　chcat：改变语境类别

作用：chcat 命令用于改变语境类别。

用法：/usr/bin/chcat CATEGORY 文件...

/usr/bin/chcat -l CATEGORY 用户...

/usr/bin/chcat [[+|-]CATEGORY],...]q File ...

/usr/bin/chcat -l [[+|-]CATEGORY],...]q 用户...

/usr/bin/chcat -d File ...

/usr/bin/chcat -l -d 用户...

/usr/bin/chcat -L

/usr/bin/chcat -L -l 用户...

主要选项如下。

- -d：删除类别。
- -L：显示可用类别。
- -l：对用户操作而不是文件。

应用实例如下。

（1）显示所有类别。

```
# chcat -L
s0
s0-s0:c0.c1023          SystemLow-SystemHigh
s0:c0.c1023             SystemHigh
```

（2）显示 root 用户类别。

```
# chcat -lL root
root: SystemHigh
```

9.11.14 restorecon：恢复文件安全语境

作用：restorecon 命令用来恢复文件安全语境。

用法：restorecon [-iFnrRv] [-e excludedir] [-o filename] [-f filename | pathname...]

主要选项如下。

- -i：忽略不存在的文件。
- -f infilename：记录要处理的文件。
- -e：directory 排除目录。
- -R -r：递归处理目录。
- -n：不改变文件标签。
- -o outfilename：在文件不正确的情况下，保存文件列表到 outfilename。
- -v：将过程显示到屏幕上。
- -F：强制恢复文件安全语境。

说明

restorecon 命令和 chcon 命令类似，但它基于当前策略默认文件上下文文件设置与文件有关的客体的安全上下文，因此，用户没有指定一个安全上下文。相反，restorecon 使用文件上下文文件的条目匹配文件名，然后应用特定的安全上下文。在某些情况下，它是在还原正确的安全上下文。

应用实例如下。

（1）将错误的 index.html 以预设的安全性本文改正过来。

```
# restorecon -Rv /var/www/html/index.html
restorecon reset /var/www/html/index.html context system_u:object_r:etc
_t:s0->
system_u:object_r:httpd_sys_content_t:s0
```

上面这两行其实是同一行，表示将 index.html 由 etc_t 改为 httpd_sys_content_t。

（2）使用 restorecon 命令检查与文件有关的客体的标记和文件上下文文件中的条目是否匹配。

```
$ mkdir public_html
$ /sbin/restorecon -nv public_html/
/sbin/restorecon reset /home/joe/public_html context
joe:object_r:user_home_dir_t->user_u:object_r:httpd_user_content_t
```

在这个例子中，我们使用-n 和-v 选项阻止 restorecon 真正地执行标记，它只输出标记改变的信息，这两个选项一起让 restorecon 输出在磁盘上的标记和文件上下文文件标记之间的差异。restorecon 命令可以用来递归标记大量的文件，-R 选项告诉 restorecon 命令递归处理目录，重新标记所有的子目录和文件。

9.11.15　chcon：改变文件安全语境

作用：chcon 命令用来改变 SELinux 安全上下文。

用法：chcon [选项] CONTEXT 文件

主要选项如下。

– -R：递归改变文件和目录的上下文。
– --reference：从源文件向目标文件复制安全上下文。
– -h, --no-dereference：影响目标链接。
– -v, --verbose：输出对每个检查文件的诊断。
– -u, --user=USER：设置在目标用户的安全上下文。
– -r,--role=ROLE：设置目标安全领域的作用。
– -t, --type=TYPE：在目标设定的安全上下文类型。
– -l, --range=RANGE：设置目标安全领域的范围。
– -f：显示少量错误信息。

说明

chcon 命令基于用户的输入为一个或多个文件设置相同的安全上下文，它是最基础的标记命令。

应用实例如下。

```
$ mkdir public_html
$ ls -dZ public_html/
drwxrwxr-xjoe joe joe:object_r:user_home_dir_t public_html/
$ chcon -t httpd_user_content_t public_html/
$ ls -dZ public_html/
drwxrwxr-xjoe joe joe:object_r:httpd_user_content_t public_html/
```

在这个例子中，我们修改了新创建的目录的安全上下文。创建时自动分配的安全上下文是 joe:object_r:user_home_dir_t，然后修改成 joe:object_r:httpd_user_content_t，-t 选项指出文件的类型应该改变，而安全上下文的剩余部分保持不动。

9.11.16 setfiles：设置文件安全语境

作用：setfiles 命令用来设置文件安全语境。

用法：setfiles[-cpolicy] [-d] [-l] [-n] [-edirectory] [-o filename] [-q] [-s] [-v] [-vv] [-W] [-F] spec_file pathname...

主要选项如下。

- -r rootpath：使用一个备用根路径。
- -e directory：排除目录。
- -F：强制设置文件安全语境。
- -o outfilename：在文件不正确的情况下，保存文件列表到 outfilename。
- -s：从标准输出读取文件列表。
- -v：如果类型或角色正在发生变化，文件标签也有所不同。
- -W：如果类型、角色或用户正在发生变化，没有匹配的文件显示警告。

应用实例如下。

重新设置文件系统的标签。

```
#setfiles file_contexts/
file_contexts/是用户定义的文件
```

9.11.17 seinfo：提取策略的规则数量统计信息

作用：seinfo 命令从规则 policy.conf 或二进制规则 policy 中提取策略的规则数量统计信息。

用法：seinfo[选项] [规则文件]

主要选项如下。

- -A：列出 SELinux 的状态、规则布尔值、身份识别、角色、类别等所有信息。
- -t：列出 SELinux 的所有类别（type）种类。
- -r：列出 SELinux 的所有角色（role）种类。

- -u：列出 SELinux 的所有身份识别（user）种类。
- -b：列出所有规则的种类（布尔值）。

应用实例如下。

（1）列出 SELinux 在此政策下的统计状态。

```
# seinfo
Statistics for policy file: /etc/selinux/targeted/policy/policy.21
Policy Version & Type: v.21 (binary, MLS)
 Classes:                61      Permissions:      220
 Types:                1710      Attributes:       161
 Users:                   3      Roles:              6
 Booleans:              242      Cond. Expr.:      222
 Sensitivities:           1      Categories:      1024
 Allow:              116810      Neverallow:         0
 Auditallow:             41      Dontaudit:       6778
 Role allow:              5      Role trans:         0
 Type_trans:           1886      Type_change:        0
 Type_member:             0      Range_trans:      317
 Constraints:            47      Validatetrans:      0
 Fs_use:                 18      Genfscon:          74
 Portcon:               323      Netifcon:           0
 Nodecon:                 8      Initial SIDs:      27
```

从上面我们可以看到这个政策是 targeted，此政策的安全性本文类别有 1521 个，而针对网络服务的规则（Booleans）共制定了 213 条。

（2）列出与 httpd 有关的规则（Booleans）。

```
#seinfo -b | grep httpd
Rule loading disabled
    allow_httpd_mod_auth_pam
   allow_httpd_bugzilla_script_anon_write
   httpd_enable_ftp_server
   httpd_use_nfs
   allow_httpd_squid_script_anon_write
   allow_httpd_anon_write
   httpd_can_network_relay
   httpd_disable_trans
   allow_httpd_prewikka_script_anon_write
   httpd_tty_comm
   httpd_unified
```

```
    httpd_rotatelogs_disable_trans
    httpd_builtin_scripting
    httpd_can_sendmail
    httpd_enable_cgi
    allow_httpd_nagios_script_anon_write
    httpd_suexec_disable_trans
    httpd_enable_homedirs
    httpd_ssi_exec
    httpd_use_cifs
    allow_httpd_cvs_script_anon_write
    allow_httpd_sys_script_anon_write
    httpd_can_network_connect
    httpd_can_network_connect_db
```

9.11.18　sesearch：搜索 policy.conf 或二进制策略中特别的类型

作用：sesearch 命令搜索 policy.conf 或二进制策略中特别的类型。

用法：sesearch [选项] [POLICY_FILE]

主要选项如下。

- -a：列出该类别或布尔值的所有相关信息。
- -t：后面还要接类别，如-t httpd_t。
- -b：后面还要接布尔值的规则，如-b httpd_enable_ftp_server。

应用实例如下。

（1）找出目标档案资源类别为 httpd_sys_content_t 的有关信息。

```
# sesearch -a -t httpd_sys_content_t
Found 74 av rules:
   allow readahead_t httpd_sys_content_t : file {ioctl read getattr lock}；
   allow readahead_t httpd_sys_content_t : dir {ioctl read getattr lock search}；
...
```

（2）找出主体程序为 httpd_t 且目标档案类别为 httpd 相关的所有信息。

```
# sesearch -s httpd_t -t httpd_* -a
Found 163 av rules:
   allow httpd_t httpd_sys_content_t : file { ioctl read getattr lock }；
   allow httpd_t httpd_sys_content_t : dir { ioctl read getattr lock search }；
   allow httpd_t httpd_sys_content_t : lnk_file { ioctl read getattr lock }；
...
```

从上面的数据就可以看出当程序为 httpd_t 这个类别时，是可以读取 httpd_sys_ content_t 的。

（3）知道有个布尔值为 httpd_enable_homedirs，查看该布尔值规范多少规则。

```
# sesearch -b httpd_enable_homedirs -a
Found 21 av rules:
 allow httpd_t user_home_dir_t : dir { getattr search };
 allow httpd_t cifs_t : file { ioctl read getattr lock };
 allow httpd_t cifs_t : dir { ioctl read getattr lock search };
...
```

从这个布尔值的设定我们可以看到，里面规范了非常多的主体程序与目标档案资源的放行与否。所以用户可以知道，实际上规范这些规则的就是布尔值的项目吗，那也就是我们之前所说的一堆规则，用户的主体程序能否对某些目标档案进行存取，与这个布尔值有很大关系，因为布尔值可以将规则设定为启动或者关闭。

9.11.19　checkmodule：编译策略模块

作用：checkmodule 命令用来编译策略模块。

用法：checkmodule [-V] [-b] [-m] [-M] [-o FILE] [INPUT]

主要选项如下。

- INPUT：从输入建立模块（否则，由"policy.conf"读模块）。
- -V：表明政策程序创建的版本。
- -b：读取二进制策略文件。
- -m：建立一个基础模块，而不是一个策略模块。
- -o FILE：写模块到文件（其他只是检查语法）。
- -M：启用 MLS 策略。

应用实例如下。

由文件 local.te 编译模块。

```
# checkmodule -m -o local.mod local.te
  checkmodule: loading policy configuration from local.te
  checkmodule: policy configuration loaded
  checkmodule: writing binary representation (version 5) to local.mod
```

9.11.20　sealert：SELinux 信息诊断客户端工具

作用：命令 sealert 是 SELinux 信息诊断客户端工具。

用法：sealert [选项]

主要选项如下。

- -H, --html_output：使用网页格式输出（搭配-a 或-l 使用）。
- -l, --lookupid ID：检视指定 ID 的警示信息。
- -b -browser：启动图形界面的信息诊断客户端工具。

- -h -help：显示帮助。
- -a --analyze file：分析日志文件。
- -u -user：使用用户登录。
- -b -password：设置用户口令。

> **说明**
>
> 命令 sealert 需要搭配 setroubleshoot 服务一起使用。setroubleshoot 服务启动后，会依据 audit 服务提供的信息给予适当问题诊断，然后输出于/var/log/messages，该档案内会有相关输出信息提供除错检视。

应用实例如下。

启动 setroubleshoot，以便可以使用 sealert 命令查询错误数据库，查询完毕后可以关闭。

```
#service setroubleshoot start
```

观察/var/log/messages 档案的末段，会发现 SELinux 的 setroubleshoot 记载的信息，可以用 sealert -l 8c613f95-0518-4548-bc01-574f9a152eb7 指令来观看完整记录。

```
#sealert -l 8c613f95-0518-4548-bc01-574f9a152eb7
```

结果如图 9-9 所示。

```
[root@localhost ~]# sealert -l 4b40a679-9e7f-494d-ba1e-cb247e78af2a
摘要:

SELinux is preventing ping (ping_t) "read write" to
/usr/local/nagios/var/spool/checkresults/checkhLyzZo (usr_t).

详细的描述:

SELinux denied access requested by ping. It is not expected that this access is
required by ping and this access may signal an intrusion attempt. It is also
possible that the specific version or configuration of the application is
causing it to require additional access.

正在允许访问:

Sometimes labeling problems can cause SELinux denials. You could try to restore
the default system file context for
/usr/local/nagios/var/spool/checkresults/checkhLyzZo,

restorecon -v '/usr/local/nagios/var/spool/checkresults/checkhLyzZo'

If this does not work, there is currently no automatic way to allow this access.
Instead, you can generate a local policy module to allow this access - see FAQ
(http://fedora.redhat.com/docs/selinux-faq-fc5/#id2961385) Or you can disable
SELinux protection altogether. Disabling SELinux protection is not recommended.
Please file a bug report (http://bugzilla.redhat.com/bugzilla/enter_bug.cgi)
against this package.

附加的信息:

源上下文                  system_u:system_r:ping_t
目标上下文                system_u:object_r:usr_t
目标对象                  /usr/local/nagios/var/spool/checkresults/checkhLyz
                              Zo [ file ]
Source                        ping
Source Path                   /bin/ping
Port                          <未知的>
Host                          localhost.localdomain
Source RPM Packages           iputils-20020927-46.el5
Target RPM Packages
策略 RPM                    selinux-policy-2.4.6-255.el5
Selinux 激活                True
策略类型                    targeted
```

图 9-9　查看错误数据库

9.11.21　selinuxenabled：查询系统的 SELinux 是否启用

作用：selinuxenabled 命令用于查询系统的 SELinux 是否启用。

用法：selinuxenabled

该命令无选项。

说明

回传结束代码为 0，表示系统打开 SELinux 功能；回传结束代码为 1，表示系统关闭 SELinux 功能。

9.11.22　SELinux 工具总结列表

这里笔者把 SELinux 工具做一下总结：SELinux 分析工具如表 9-6 所示，SELinux 策略建立工具如表 9-7 所示，SELinux 调试工具如表 9-8 所示，SELinux 开发工具如表 9-9 所示，SELinux 重置工具如表 9-10 所示，SELinux 运行工具如表 9-11 所示。

表 9-6　SELinux 分析工具

工具名称	功能描述
apol	apol 是基于 GUI 图像界面的策略分析工具。分析 SELinux 策略，帮助发现策略中类型之间的关系。策略分析是设计策略的重要环节。在策略装载到内核之前，apol 能帮助发现策略决策将产生的不期望的结果，还可帮助显示不同版本或种类的策略之间的差异
sediff	执行两个策略差别分析
sediffx	图形化版本的 sediff
sesearch	搜索 policy.conf 或二进制策略中特别的类型

表 9-7　SELinux 策略建立工具

工具名称	功能描述
checkmodule	编译策略模块
checkpolicy	从 policy.conf 文件加载策略
sedoctool.py	建立策略文档
semodule_package	创建策略模块包

表 9-8　SELinux 调试工具

工具名称	功能描述
avcstat	avcstat 提供了从启动以来的访问向量缓存（Access Vector Cache，AVC）的统计信息，能指定时间间隔实时查看 AVC 的信息
chkcon	检查 SELinux 安全上下文

续表

工具名称	功能描述
getfilecon	检查上下文路径
getpidcon	检查上下文进程
getseuser	检查用户上下文登录
policyvers	根据策略文件检查内核安全
seinfo	从规则 policy.conf 或二进制规则 policy 中提取策略的规则数量统计信息
selinux_check_secure_tty_context	检查 tty 上下文是否是一个安全上下文
sestatus	sestatus 可查询 SELinux 的状态，如是否激活、selinuxfs 的挂接状态、当前的策略模式和版本等

表 9-9　SELinux 开发工具

工具名称	功能描述
audit2allow	生成策略允许规则
audit2why	转换成一个 SELinux 的审计信息描述为何访问被拒绝
CDS Framework	编写的策略 IDE 工具
indexcon	建立数据库上下文
matchpathcon	查询活动上下文如何进行路径标定
seaudit	基于 GUI 图像界面的工具 seaudit 是 SELinux 的 Log 信息分析工具，它可让用户排序和过滤 Log 信息，或输出规范格式的报告，方便用户阅读或分析 Log 信息

表 9-10　SELinux 重置工具

工具名称	功能描述
chcon	改变 SELinux 安全上下文
fixfiles	当转换不同的策略或使用一个新文件系统时，需要标识整个文件系统。转换策略时，应该重新启动机器让 init 进程完成重标识工作，因为不重新启动机器，很难让所有的进程得到正确的标识，特别是后台进程。使用 fixfiles 命令重标识文件系统或目标，还可恢复 RPM 包的安全上下文
restorecon	恢复文件安全语境
restorecond	restorecon 命令守护进程
setfilecon	设置上下文文件路径
setfiles	设置文件安全语境

表 9-11　SELinux 运行工具

工具名称	功能描述
chcat	改变语境类别
getenforce	查看 SELinux 模式
getsebool	查看 SELinux 布尔值
load_policy	装载策略

续表

工具名称	功能描述
sealert	SELinux 信息诊断客户端工具
segatex	第三方图形化工具
selinuxenabled	查询系统的 SELinux 是否启用
semanage	把 Linux 用户名捆绑到 SELinux 用户
semodule	管理策略模块
setenforce	设置 SELinux 模式
setroubleshootd	SELinux 故障排除工具
setsebool	设置 SELinux 布尔值
runcon	用指定的安全上下文运行
togglesebool	转换内核中的布尔值到反方向值，如从 1 到 0。不改动/etc/selinux/targeted/booleans 文件中的值
newrole	newrole 表示用指定的类型或角色运行新的 Shell 脚本。在 targeted 策略中，角色一般都为 system_r，角色转换的意义不大；而在 strict 策略中角色较多，角色转换作用较大。如果研发测试新类型，转换角色也是非常有用的
system-config-selinux	Red Hat 图形化管理工具
togglesebool	翻转 SELinux 布尔值
update-selinux-conf	升级 SELinux 配置文件/etc/selinux/config（Ubuntu 平台）
update-selinux-policy	升级 SELinux 策略文件/etc/selinux.d（Ubuntu 平台）

9.12 虚拟化管理命令

9.12.1　xen 虚拟化管理命令

1．xm：xen 用户管理接口

作用：xm 命令是 xen 用户管理接口工具，这个程序可以建立、暂停和关闭 domains。当然，也可以列出当前的 domains，固定、激活 vcpus，添加或者移除虚拟块设备。

用法：xm <子命令> [ARGS]

每个 xm 命令的基本格式如下：

```
xm <子命令> <domain-id> [选项]
```

domain-id 是 domain 的 ID 或者 domain 的名字（domain 会给解释成 domain-id），选项是子命令的特定选项。

xm 常用子命令如下。

- console：连接至虚拟机的控制台。
- create：建立一个 domain-u 的虚拟机。

- destroy：删除一个 domain-u 的虚拟机。
- list：所有已知的虚拟机列表。
- mem-set：重新设置 domain-u 的内存大小。
- pause：暂停虚拟机的活动而不释放使用的内存资源。
- reboot：重新启动某一台虚拟机。
- restore：还原虚拟机。
- save：存储虚拟机的状态。
- shutdown：关闭虚拟机。
- top：监控虚拟机的执行状态。
- unpause：取消暂停，即继续执行虚拟机。
- uptime：显示虚拟机的 uptime。
- vcpu-set：修改虚拟机的 CPU 数量。
- help：所有可用的命令概述。
- top：提供所有虚拟机的状态概貌。
- console：打开控制台管理虚拟机。
- new：添加虚拟机到 Xenbase 托管环境。
- start：从 Xenbase 托管环境启动虚拟机。
- destroy：像关掉电源那样关闭虚拟机。
- reboot：重新启动虚拟机。
- vnet-list：所有虚拟机网络接口列表。
- vnet-create：从配置文件创建一个虚拟机网络接口。
- vnet-delete：删除虚拟机网络接口。
- info：获取 xen 主机信息。
- log：打印 xend 日志。
- network-detach：终止用户所指定的虚拟网络设备。
- addlabel：创建一个域安全性标签。
- block-attach：创建一个新的虚拟块设备。
- block-detach：销毁一个现有的虚拟块设备。
- rmlabel：删除一个域安全性标签。
- makepolicy：创建一个二进制策略。
- loadpolicy：加载一个二进制策略。
- labels：查看策略标签。
- getlables：查看域的安全性标签。
- mem-max：设置 domain 使用的最大内存。
- mem-set：用来修改虚拟机内存的目前使用量。值得注意的是，mem-set 设置的值不能超过 mem-max 设置的值的大小。
- migrate：迁移 domain 到另外一台计算机。
- rename：给域分配一个新的名字。

– domname：把 domain ID 转换为域名。
– vtpm-list：显示虚拟 TPM 设备。

主要 ARGS 如下。

– -c：在 domain 创建的同时装在控制台。这个选项在 domain 崩溃的时候很有用。该选项允许用户连接至文本控制台来协助解除故障。
– -a：关闭系统里的所有域。
– -w：用于等待域的完全关闭。
– count：必须指定 count（以 MB 为单位）。
– vcpu：希望指定的 vcpu，cpus 是目标。
– --long：提供用户所指定的域的更详细的信息。
– --label：添加了一个字段来显示 label 状态。
– policy：二进制策略，保存为二进制文件。

说明

不同的子命令搭配的 ARGS 也会不相同。

应用实例如下。

1）显示域状

可以使用 xm list --label 来显示一个或多个域的活动状态。

```
# xm list --label
Name                              ID Mem(MiB) VCPUs State   Time(s) Label
Domain-0                           0     2447     2 r-----    209.3  INACTIVE
cjh                                1      512     1 -b----     80.2  INACTIVE
```

显示区域时，xm 会显示下列的每一个字段。

– Name：这个字段显示虚拟机的名称。
– ID：这是虚拟机的标识符。
– Mem：虚拟机目前使用多少内存。
– VCPUs：虚拟机目前使用的 CPU 数量。
– State：虚拟机的状态。在这个字段中，若显示若干个标签，则每个标签代表的意义如表 9-12 所示。
– Time：虚拟机执行共花费多少时间。

表 9-12　域状态

状　态	描　述
running	列出当前在某个 CPU 上活动的域
blocked	列出被阻塞的域（当 vcpu 在等待外部事件的发生时，域将被阻塞）
paused	列出被挂起的域

续表

状　态	描　述
shutdown	列出正被关闭的域
shutoff	列出已被完全关闭的域
crashed	列出崩溃的域
inactive	列出是不活动实例的域
-all	列出是活动或不活动的 vcpu 实例的域

2）连接至域

可以使用 xm 来连接至域或虚拟机。

```
#xm console cjh1
```

这使控制台附加到 domain-id 的文本控制台。

3）创建一个域

```
#xm create cjh
Using config file "./cjh".
Started domain cjh
```

4）监控一个域

如果想要监控每一台虚拟机的状态，则可以使用 xen top，如图 9-10 所示。

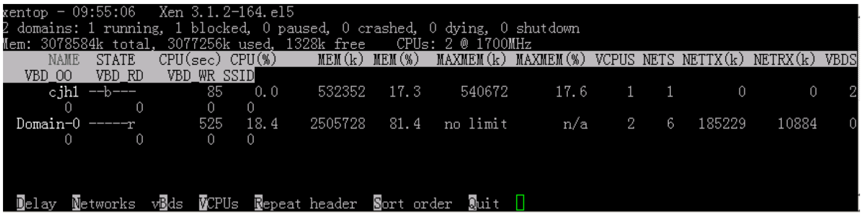


图 9-10　监控一个域

在监控虚拟机的过程中，你会看到画面的下方有一行信息，提示你可以做什么动作。在监控的过程中可以执行的动作整理如表 9-13 所示。

表 9-13　在监控中可以执行的动作

项　目	按　键	说　明
Delay	D	修改显示更新频率，单位为秒
Networks	N	修改显示网络的状态
vBds	B	切换是否显示虚拟磁盘的状态
VCPUs	V	切换是否显示虚拟中央处理器信息
Repeat header	R	切换是否在每一台虚拟机前显示表格字段
Sort order	S	切换排序的字段
Quit	Q	结束 xm top，回到 Shell

5）修改 Domain-U

当然，与虚拟系统管理一样，也可以利用 xm 来修改 Domain-U 的硬件设备的设置。可以执行如下操作调整 cjh1 的内存大小为 256MB。

```
# xm mem-set cjh1 256
```

Xen 并不会马上修改，它会慢慢地调降至指定的大小。稍候片刻，system1 的内存就变成了 256MB。

6）挂起域

可以用 xm 来挂起域。

```
#xm suspend cjh1
```

7）显示运行时间

可以用 xm 来显示运行时间。

```
# xm uptime cjh1
Name                       ID  Uptime
cjh1                       5   0:20:24
```

8）用 xm 来显示域的 CPU 信息

必须指定要列出哪些 vcpu。如果没有指定，所有域的 vcpu 都将被列出。

```
# xm vcpu-list
Name                             ID VCPUs   CPU State   Time(s) CPU Affinity
Domain-0                          0     0     0  r--     383.7 0
Domain-0                          0     1     1  r--     296.5 1
cjh1                              5     0     0  -b-     95.0 any cpu
```

9）显示域信息

可以使用 xm info 来显示主机域信息。

```
# xm info
host                       : localhost.localdomain
release                    : 2.6.18-164.el5xen
version                    : #1 SMP Thu Sep 3 04:47:32 EDT 2009
machine                    : i686
nr_cpus                    : 2
nr_nodes                   : 1
sockets_per_node           : 1
cores_per_socket           : 2
threads_per_core           : 1
cpu_mhz                    : 1700
hw_caps
```

```
178bfbff:ebd3fbff:00000000:00000010:00002001:00000000:0000011f
total_memory           : 3006
free_memory            : 1
node_to_cpu            : node0:0-1
xen_major              : 3
xen_minor              : 1
xen_extra              : .2-164.el5
xen_caps               : xen-3.0-x86_32p hvm-3.0-x86_32 hvm-3.0-x86_32p
xen_pagesize           : 4096
platform_params        : virt_start=0xf5800000
xen_changeset          : unavailable
cc_compiler            : gcc version 4.1.2 20080704 (Red Hat 4.1.2-46)
cc_compile_by          : mockbuild
cc_compile_domain      : centos.org
cc_compile_date        : Thu Sep3 03:26:12 EDT 2009
xend_config_format     : 2
```

2．virsh：命令模式的 Xen 管理工具

作用：virsh（Virtual Shell）是由一个名为 libvirt 的软件提供的管理工具，它提供管理虚拟机更高级的能力。

用法：virsh <子命令>[ARGS]

virsh 提供两种执行模式：直接模式（Direct Mode）与互动模式（Interactive Mode）。在直接模式里，必须在 Shell 中以参数、自变量的方式来执行 virsh；如果在互动模式中，则 virsh 会提供一个提示字符串，可以在该提示字符串后输入要执行的命令。如果执行 virsh 没有指定任何参数或自变量，则默认进入互动模式。

主要子命令如下。

- Help：打印帮助。
- attach-device：从一个 XML 文件附加装置。
- attach-disk：附加磁盘设备。
- attach-interface：获得网络界面。
- autostart：自动开始一个域。
- capabilities：性能。
- connect：连接（重新连接）到 Hypervisor。
- console：连接到客户会话。
- create：从一个 XML 文件创建一个域。
- start：开始一个（以前定义的）非活跃的域。
- destroy：删除一个域。
- detach-device：从一个 XML 文件分离设备。
- detach-disk：分离磁盘设备。

- detach-interface：分离网络界面。
- define：从一个 XML 文件定义（但不开始）一个域。
- domid：把一个域名或 UUID 转换为域 ID。
- domuuid：把一个域名或 ID 转换为域 UUID。
- dominfo：域信息。
- domname：将域 ID 或 UUID 转换为域名。
- domstate：域状态。
- domblkstat：获得域设备块状态。
- domifstat：获得域网络接口状态。
- dumpxml：XML 中的域信息。
- edit：编辑某个域的 XML 配置。
- find-storage-pool-sources：发现潜在存储池源。
- find-storage-pool-sources-as：找到潜在存储池源。
- freecell：NUMA 可用内存。
- hostname：打印管理程序主机名。
- list：列出域。
- migrate：将域迁移到另一台主机中。
- net-autostart：自动开始网络。
- net-create：从一个 XML 文件创建一个网络。
- net-define：从一个 XML 文件定义（但不开始）一个网络。
- net-destroy：删除一个网络。
- net-dumpxml：XML 中的网络信息。
- net-edit：为网络编辑 XML 配置。
- net-list：列出网络。
- net-name：把一个网络 UUID 转换为网络名。
- net-start：开始一个（以前定义的）不活跃的网络。
- net-undefine：取消定义一个非活跃的网络。
- net-uuid：把一个网络名转换为网络 UUID。
- nodeinfo：节点信息。
- nodedev-list：这台主机中的枚举设备。
- nodedev-dumpxml：XML 中的节点设备详情。
- nodedev-dettach：分离节点设备及其设备驱动程序。
- nodedev-reattach：重新附加节点设备驱动程序。
- nodedev-reset：重置节点设备。
- pool-autostart：自动启动某个池。
- pool-build：建立池。
- pool-create：从一个 XML 文件中创建一个池。
- pool-create-as：从一组变量中创建一个池。

- pool-define：在一个 XML 文件中定义（但不启动）一个池。
- pool-define-as：在一组变量中定义池。
- pool-destroy：销毁池。
- pool-delete：删除池。
- pool-dumpxml：XML 中的池信息。
- pool-edit：为存储池编辑 XML 配置。
- pool-info：存储池信息。
- pool-list：列出池。
- pool-name：将池 UUID 转换为池名称。
- pool-refresh：刷新池。
- pool-start：启动一个（以前定义的）非活跃的池。
- pool-undefine：取消定义一个不活跃的池。
- pool-uuid：把一个池名称转换为池 UUID。
- quit：退出这个非交互式终端。
- reboot：重新启动一个域。
- restore：从一个文件中的状态恢复一个域。
- resume：重新恢复一个域。
- save：把一个域的状态保存到一个文件中。
- schedinfo：显示/设置日程安排变量。
- dump：把一个域的内核 dump 到一个文件中以方便分析。
- shutdown：关闭一个域。
- setmem：改变内存的分配。
- setmaxmem：改变最大内存限制值。
- setvcpus：改变虚拟 CPU 的号。
- suspend：挂起一个域。
- ttyconsole：tty 控制台。
- undefine：取消定义一个非活跃的域。
- uri：打印管理程序典型的 URI。
- vol-create：从一个 XML 文件创建一个卷。
- vol-create-as：从一组变量中创建卷。
- vol-delete：删除卷。
- vol-dumpxml：XML 中的卷信息。
- vol-info：存储卷信息。
- vol-list：列出卷。
- vol-path：将卷 UUID 转换为卷路径。
- vol-name：将卷 UUID 转换为卷名称。
- vol-key：将卷 UUID 转换为卷密钥。
- vcpuinfo：域 vcpu 的信息。

– vcpupin：控制域 vcpu affinity。
– version：显示版本。
– vncdisplay：vnc 显示。

irsh 通过主命令和子命令管理每个虚拟机功能。要查看所有命令，输入 virsh help。你可能发现大量命令，不过每个命令都有帮助指南，只需要输入 virsh help vol-delete，就可以看到所有关于 virsh vol-delete 命令的帮助。要学习 virsh 很困难，因为有许多命令。不过一旦你获得一个很强的命令功能，就能获得收益。

而 virsh 命令语法的 ARGS 则是子命令搭配的参数。不同的子命令可能需要搭配不同的参数。总的来说，如果 ARGS 需指定虚拟机，则可以使用下列格式指定是哪一台虚拟机。

– 虚拟机的名称。
– 虚拟机启动后的标识符。
– 虚拟机的 UUID。

说明

virsh 是由一个名为 libvirt 的软件提供的管理工具，提供管理虚拟机更高级的能力。virsh 大部分的功能与 xm 一样，可以利用 virsh 来启动、删除、控制、监控 Xen 的区域，因此也可以利用 virsh 来管理 Xen 中所有的虚拟机。你可能会很好奇，既然已经提供了 xm 管理工具，为何 Linux 还需要提供 virsh？这是因为在 Linux 领域里，除了 Xen 以外，还有许多种机制也提供虚拟化的功能，然而，不同的虚拟化系统的使用方法都不太相同。为了让 Linux 可以通过同一种方法读取与管理各种虚拟化子系统，libvirt 团队（http://libvirt.org/）便开发出了 libvirt.so 链接库，让软件开发人员可以用 libvirt.so 提供的 API 来管控所有的虚拟化系统。而 virsh 就是利用 libvirt.so 链接库编写而成的管理工具。因此，不管实质上执行的是何种虚拟化子系统，你只需学会 virsh 的使用方法，就可以借此来使用或管理各种虚拟化系统提供的虚拟机了。libvirt 为表 9-14 所列的虚拟机监控程序实现了驱动程序。随着新的虚拟机监控程序在开源社区出现，其他驱动程序无疑也将可用。

表 9-14　libvirt 支持的虚拟机监控程序

虚拟机监控程序	描　述
Xen	面向 IA-32、IA-64 和 PowerPC 970 架构的虚拟机监控程序
QEMU	面向各种架构的平台仿真器
Kernel-based Virtual Machine（KVM）	Linux 平台仿真器
Linux Containers（LXC）	用于操作系统虚拟化的 Linux（轻量级）容器
OpenVZ	基于 Linux 内核的操作系统级虚拟化
VirtualBox	x86 虚拟化虚拟机监控程序
User Mode Linux	面向各种架构的 Linux 平台仿真器
Test	面向伪虚拟机监控程序的测试驱动器
Storage	存储池驱动器（本地磁盘、网络磁盘、iSCSI 卷）

应用实例如下。

1）查看 Domain 信息

virsh 可以查看虚拟机的相关信息，只需在执行 virsh 命令时配合如表 9-15 所示的命令。

表 9-15　查看 Domain 信息执行的命令

命　令	说　明
domid	查看虚拟机的标识符
domname	查看虚拟机的名称
domuuid	查看虚拟机的 UUID
domstate	查看虚拟机目前的状态
dominfo	查看虚拟机的信息

执行上述每一条命令时，都必须指定虚拟机的名称、标识符和 UUID。如图 9-11 所示为使用 virsh 查看虚拟机信息的示范步骤。

```
[root@localhost ~]# virsh domid cjh1
5

[root@localhost ~]# virsh domname 5
cjh1

[root@localhost ~]# virsh domuuid cjh1
30b7d0ce-3828-abd9-8fa2-85649e60d7fc

[root@localhost ~]# virsh dominfo cjh1
Id:             5
名称:         cjh1
UUID:           30b7d0ce-3828-abd9-8fa2-85649e60d7fc
OS 类型:      hvm
状态:         闲置
CPU:            1
CPU 时间:     132.7s
最大内存:   540672 kB
使用的内存:  532352 kB
自动启动:   禁用
```

图 9-11　使用 virsh 查看虚拟机信息的示范步骤

2）显示虚拟机信息

可以用 virsh 来显示虚拟机列表和当前状态。

```
virsh list domain-name [——inactive|—— -all]
```

- ——inactive 选项列出不活动的域（已经被定义但当前不活动的域）。
- ——-all 选项列出所有的域，包括活动和不活动的。

输出和下例类似：

```
# virsh list
 Id    名称     状态
----------------------------------
0    Domain-0    running
5    cjh1        闲置
```

3）配置内存数量

可以用 virsh 来修改某个域的内存分配。

```
#virsh setmem [domain-id | domain-name] [count]
```

你必须指定[count]（以 KB 为单位）。注意，新的值不能超过创建虚拟机时指定的数量，低于 64MB 的值可能无法工作。可以根据需要调整虚拟机的内存。

下面是“互动模式（Interactive Mode）”下命令的使用方法：

```
 virsh # setmem cjh1 524288
```

下面是“直接模式（Direct Mode）”下命令的使用方法：

```
# virsh setmem Domain-0 $(( 1024 * 512 ))
```

4）显示虚拟 CPU 信息

可以用 virsh vcpuinfo 来显示虚拟机里的虚拟 CPU 信息。

```
# virsh vcpuinfo cjh1
VCPU:        0
CPU:         0
状态:        闲置
CPU 时间:    145.4s
CPU 关系:    yy
```

3．xentop：实时显示 Xen 主机系统信息

作用：xentop 命令用于实时显示 Xen 主机系统信息。

用法：xentop[选项]

主要选项如下。

- -h, --help：显示帮助。
- -V, --version：输出版本信息。
- -d, --delay=SECONDS：定时更新（默认是 3 秒）。
- -n, --networks：网络数据输出。
- -v, --vcpus：vcpu 数据输出。
- -b, --batch：批处理数据输出。
- -i, --iterations：退出之前反复迭代次数。

> **说明**
>
> 使用 Linux 的人都很熟悉“xentop”命令，因为它的名字和传统的“top”命令很相似。这个命令实时显示服务器的性能数据和主机中虚拟机的性能数据。xentop 使用的是虚拟机的 UUID（Universally Unique Identifier），而不是更有用的虚拟机名。xentop 可以自由打印 UUID 和虚拟机名，不过虚拟机名是必须打印的。

应用实例如下。

想要在 Xen 上获取某台虚拟机的 CPU 使用率，于是使用命令 xentop -b -i 1，获得类似下面的数据：

```
    # xentop -b -i 1
    xentop - 11:26:48   Xen 3.1.2-164.el5
    2 domains: 1 running, 1 blocked, 0 paused, 0 crashed, 0 dying, 0 shutdown
    Mem: 3078584k total, 3077560k used, 1024k free    CPUs: 2 @ 1700MHz
    NAME  STATE   CPU(sec) CPU(%)     MEM(k) MEM(%)  MAXMEM(k) MAXMEM(%) VCPUS
NETS NETTX(k) NETRX(k) VBDS   VBD_OO   VBD_RD   VBD_WR SSID
    cjh1 --b---       172    0.0     532352   17.3     540672      17.6     1    1
0       0    2       0       0       0    0
    Domain-0 -----r      1565    0.0    2506032   81.4   no limit       n/a     2
6   421431    15709    0       0       0       0    0
```

4．virt-install：在文本模式中执行的命令

作用：在文本模式中执行命令。

用法：virt-install --name NAME -ram RAM STORAGE INSTALL[选项]

主要选项如下。

- -h, --help：显示帮助信息。
- --connect=CONNECT：使用 URI 连接到监控程序。

通用选项如下。

- -n NAME, --name=NAME：客户端事件名称。
- -r MEMORY, --ram=MEMORY：以 MB 为单位为客户端事件分配的内存。
- --arch=ARCH：模拟的 CPU 架构。
- -u UUID, --uuid=UUID：客户端 UUID。
- --vcpus=VCPUS：配置客户端的 vcpu 数目。
- --check-cpu：检查确定 vcpu 没有超过物理 CPU 数目，如果超过则发出警告。
- --cpuset=CPUSET：设定可以使用的物理 CPU。
- --os-type=DISTRO_TYPE：全虚拟客户端的操作系统类型，如 Linux、UNIX、Windows。
- --os-variant=DISTRO_VARIANT：全虚拟客户端的操作系统变体，例如：'fedora6'、'rhel5'、'solaris10'、'win2k'
- --host-device=HOSTDEVS：物理主机域名。

全虚拟化选项如下。

- --sound：使用仿真声音设备。
- --noapic：为全虚拟客户端禁用 APIC（在 os-type/os-variant db 中覆盖数值）。
- --noacpi：为全虚拟客户端禁用 ACPI（在 os-type/os-variant db 中覆盖数值）。

虚拟化类型选项如下。

- -v, --hvm：客户端应该是一个全虚拟客户端。

- -p, --paravirt：这个客户端是一个半虚拟客户端。
- --accelerate：使用内核加速功能（如 kvm、kqemu 等）。

安装方法选项如下。

- -c CDROM, --cdrom=CDROM：光驱安装介质。
- -l LOCATION, --location=LOCATION：安装源（如 nfs:host:/path、http://host/path、ftp://host/path）。
- --pxe：使用 PXE 协议从网络引导。
- --import：从一个磁盘文件导入客户机。
- --livecd：使用一个 Live CD 作为 CD-ROM。
- -x EXTRA, --extra-args=EXTRA：附加到使用--location 引导的内核的参数。

存储配置如下。

- --disk=DISKOPTS：使用不同的选项指定作为磁盘使用的存储。
- -f FILE_PATH, --file=FILE_PATH：作为磁盘映象使用的文件。
- -s DISKSIZE, --file-size=DISKSIZE：磁盘映象大小（如果它不存在的话），以 GB 为单位。
- --nonsparse：不要为磁盘使用稀疏文件。请注意这会使得客户端创建变得极其缓慢。
- --nodisks：不要为该客户端设置任何磁盘。

联网配置如下。

- -b BRIDGE, --bridge=BRIDGE：连接客户端 NIC 的桥接；如果没有给定，将尝试确定默认桥接。
- -w NETWORK, --network=NETWORK：将客户端连接到虚拟网络，使用 NAT 转发到物理网络。
- -m MAC, --mac=MAC：客户端的固定 MAC 地址；如果没有则使用随机给定的地址。
- --nonetworks：不为客户机建立网络接口。

图形配置如下。

- --vnc：用 VNC 进行图形支持。
- --vncport=VNCPORT：VNC 使用的端口。
- --sdl：用 SDL 进行图形支持。
- --nographics：不要为该客户端设置图形控制台。
- --noautoconsole：不要自动尝试连接到客户端控制台。
- -k KEYMAP, --keymap=KEYMAP：为图形控制台设置 keymap。

其他选项如下。

- -d, --debug：输入故障排除信息。
- --noreboot：安装完成时请禁用自动重启。
- --wait=WAIT：等待时间（以分钟为单位）。
- --force：对任意应用程序提示强制回答“yes”，终止所有其他提示。

应用实例如下。

（1）用命令行来创建一台 RHEL 5.4 虚拟机。

```
   # virt-install -n system1 -r 2048 -f /var/lib/xen/images/rhel5pv.dsk -s 3
--vnc -p -l ftp://10.0.0.2/ftp/dist/redhat/alphas/rhel5.4a1/Server/ia64/
```

参数说明如下。

- -n system1：命名。
- -r 2048：分配的内存大小。
- -f /var/lib/xen/images/rhel4pv.dsk：虚拟机的文件路径。
- -s 3：虚拟机的文件大小。
- --vnc：使用 vnc。
- -p：半虚拟。
- -l ftp://10.0.0.2/ftp/dist/redhat/alphas/rhel5.4a1/Server/ia64/：安装树路径或者系统 ISO。

查看一下。

```
# xm list
Name                                    ID Mem(MB) VCPUs State   Time(s)
Domain-0                                 0   30624     8 r-----    367.0
system1                                  1    2048     1 -b----      5.9
```

（2）用命令行来创建一台 Centos 4.7 虚拟机。

```
   virt-install -n firstxen -r 256 -p --nographic -f
/var/lib/xen/images/firstxen.img -s 10G -l
http://centos.ustc.edu.cn/centos/4.7/os/i386 -x "ip=192.168.1.2
netmask=255.255.255.0 dns=192.168.1.1 gateway=192.168.1.1"
```

相关解释如下。

- -n firstxen：所创建的虚拟机的名字。
- -r 256：虚拟机的内存。
- -p --nographic：两个参数是连用的，意思是非图形化安装。
- -s 10G：指定 img 大小。
- -l：指定安装介质，这里使用了一个 Centos 的 http 地址。
- -x：给虚拟机配置好 IP 信息，静态配置 IP 地址。

查看一下。

```
# xm list
Name                                    ID Mem(MB) VCPUs State   Time(s)
Domain-0                                 0   30624     8 r-----    367.0
firstxen                                 1     256     1 -b----      5.9
```

9.12.2　KVM/Qemu 虚拟机管理命令

1．Virsh

KVM 也使用 Virsh 命令格式。

应用实例如下。

（1）创建一台名为 spc 的 Debian 虚拟机，使用如下命令：

```
    # virt-install --connect qemu:///system -n spc -r 512 --vcpus=2 -f ~/vm
10.qcow2 -s 12 -c ~/debian-500-amd64-netinst.iso --vnc --noautoconsole --os
-type linux --os-variant debianlenny --accelerate --network=bridge:br0 --hv
mStarting install...
    Creating storage file...                          |  12 GB   00:00
    Creating domain...                                |  0 B   00:00
    Domain installation still in progress. You can reconnect to
    the console to complete the installation process.
```

（2）查看虚拟机。

```
    # virsh -c qemu:///system list
     Id      名称      状态
    ----------------------------------
    1   spc          running
```

2．virt-image：从一个 XML 文件建立虚拟机

作用：virt-image 命令从一个 XML 文件建立虚拟机。

用法：virt-image[选项]IMAGE.XML

主要选项如下。

- -h, --help：显示帮助。
- --connect=CONNECT：使用 URI 连接到监控程序。

通用选项如下。

- -n NAME, --name=NAME：客户端事件名称。
- -r MEMORY, --ram=MEMORY：以 MB 为单位为客户端事件分配的内存。
- -u UUID, --uuid=UUID：客户端 UUID。
- --vcpus=VCPUS：配置客户端的 vcpu 数目。
- --check-cpu：检查确定 vcpu 没有超过物理 CPU 数目，如果超过则发出警告。
- --cpuset=CPUSET：设定可以使用的物理 CPU。
- --os-type=DISTRO_TYPE：操作系统安装类型，比如 Linux、UNIX、Windows。
- --os-variant=DISTRO_VARIANT：操作系统变体安装类型，例如 fedora6、solaris10、win2k。

联网配置如下。

- -b BRIDGE,--bridge=BRIDGE：连接客户端 NIC 的桥接；如果没有给定，将尝试确定默认桥接。
- -w NETWORK, --network=NETWORK：将客户端连接到虚拟网络，使用 NAT 转发到物理网络。
- -m MAC, --mac=MAC：客户端的固定 MAC 地址；如果没有则使用随机给定的地址。

图形配置如下。

- --vnc：用 VNC 进行图形支持。
- --vncport=VNCPORT：VNC 使用的端口。
- --sdl：用 SDL 进行图形支持。
- --nographics：不要为该客户端设置图形控制台。
- -k KEYMAP, --keymap=KEYMAP：为图形控制台设置 keymap。

其他选项如下。

- -p, --print：输出 libvirt XML，但不启动该域。
- --boot=BOOT：要使用从零开始的引导记录索引。
- --replace：使用同样的名称覆盖或者销毁当前映象。
- --skip-checksum：跳过验证过程。
- -d, --debug：输入故障排除信息。
- --prompt：要求用户输入必要的选项内容。
- --force：不要提示输入。在需要的地方回答 yes，终止所有其他提示。

应用实例如下。

使用 XML 文件建立一台虚拟机。

首先建立虚拟机使用的磁盘大小为 20GB：

```
# mkdir isos
#dd if=/dev/zero of= isos/systemrescuecd.iso count bs=512k count=20480
```

这个命令需要搭配一个 XML 文件，除非你想用在大量部署客户机上面。范例内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<image>
  <name>sysresccd</name>
  <domain>
    <boot type="hvm">
      <guest>
        <arch>i686</arch>
      </guest>
      <os>
        <loader dev="cdrom"/>
      </os>
      <drive disk="root.raw" target="hda"/>
```

```
        <drive disk="sysresc"/>
      </boot>
      <devices>
        <vcpu>1</vcpu>
        <memory>262144</memory>
        <interface/>
        <graphics/>
      </devices>
    </domain>
    <storage>
      <disk file="root.raw" use="scratch" size="100" format="raw"/>
      <disk id="sysresc" file="isos/systemrescuecd.iso"
            use="system" format="iso"/>
    </storage>
  </image>
```

内容很简单，解释如下。

```
<arch>i686</arch>
Platform i686 / x86_64 或是 ppc
<loader dev="cdrom"/>
```

可以安装的介质如下。

```
<drive disk="root.raw" target="hda"/>
<drive disk="sysresc"/>
<vcpu>1</vcpu>
```

指定 vcpu 个数。

```
<memory>262144</memory>
```

指定多少内存空间给客户机，单位为 KB。

```
<disk file="root.raw" use="scratch" size="100" format="raw"/>
<disk id="sysresc" file="isos/systemrescuecd.iso" use="system" format=
"iso"/>
```

开始安装。

```
#virt-image --vnc image.xml
```

3．virt-manager：在桌面使用的虚拟化管理工具

作用：virt-manager 是在桌面使用的虚拟化管理工具。

用法：virt-manager[选项]

主要选项如下。

- -h, -help：显示帮助。
- --connect=CONNECT：使用 URI 连接到监控程序。
- --show-domain-creator：显示虚拟机创建向导。
- --show-host-summary：主窗口显示所有虚拟机性能。

应用实例如下。

在桌面启动虚拟化管理工具。

```
# virt-manager
```

Linux 桌面会出现图形化虚拟化的管理工具，如图 9-12 所示。

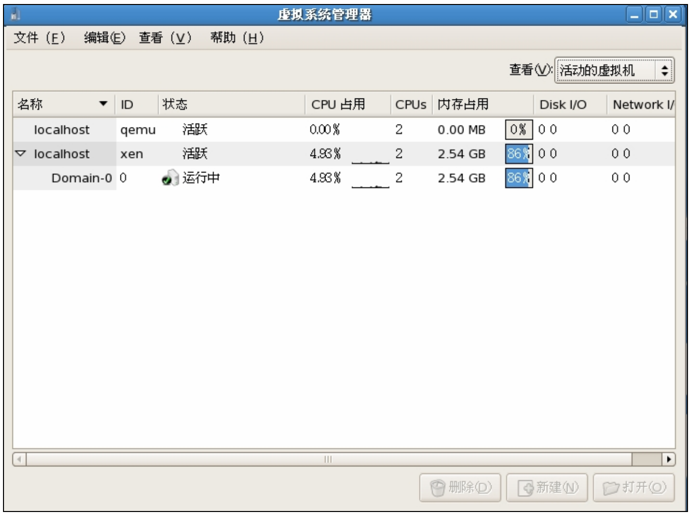


图 9-12　图形化虚拟化管理工具

4．qemu-img：创建磁盘映象

作用：qemu-img 命令用来创建磁盘映象。

用法：qemu-img 命令[命令选项]

命令如下。

```
create [-e] [-b base_image] [-f fmt] filename [size]'
commit [-f fmt] filename'
convert [-c] [-e] [-f fmt] filename [-O output_fmt] output_filename'
info [-f fmt] filename
```

主要选项如下。

- filename：磁盘映象文件名。
- base_image：只读的磁盘映象，可以作为复制到写映象的基础。写映象上的副本只存储修改的数据。
- fmt：磁盘映象格式。在大多数情况下可以自动检测，可以支持下面的格式。

- Ø raw：RAW 磁盘格式（默认）。这种格式简单并且易于移植到其他模拟器。如果用户的文件系统支持 holes（例如在 Linux 上的 Ext2 或 Ext3），而且写入的部分保持空白，可以使用 qemu-img info 来得到映象使用的实际大小或在 UNIX/Linux 上使用 ls -ls。
- Ø qcow：QEMU 映象格式，是最通用的格式。使用它可以获得较小的映象（如果用户的文件系统不支持 holes，例如在 Windows 上，这是相当有用的），可以选用 AES 加密或基于 zlib 的压缩。
- Ø cow：在写映象格式上的用户模式的 Linux 副本。在 QEMU 中作为增长的映象格式使用。这个选项只是为了与以前版本兼容，并不能在 Win32 上使用。
- Ø vmdk：VMware 3/4 兼容的映象格式。
- Ø cloop：Linux 压缩的循环映象，重用直接压缩的 CD-ROM 映象。

– size：以 K 表示的磁盘映象的尺寸。同时可以支持 M 或 G 作为前缀。
– output_filename：目的磁盘映象文件名。
– output_fmt：目标格式。

- Ø -c：表明目标映象必须是压缩的（只是 qcow 格式）。
- Ø -e：表明目标映象必须是加密的（只是 qcow 格式）。

> **说明**
>
> 使用 qemu 需要提供操作类型（create 创建新磁盘映象）、格式（qcow 用于 qemu 映象格式）、大小和磁盘映象的名称。

应用实例如下。

（1）创建一个磁盘映象。

```
#qemu-img create myimage.img mysize
```

这里 myimage.img 是磁盘映象的文件名，而 mysize 是以 K 表示的尺寸。我们可以使用 M 前缀来使用 M 表示尺寸或使用 G 前缀来使用 G 表示尺寸。

（2）生成 200GB 的磁盘映象文件。

一开始，该文件占用主机很小的磁盘空间（只有 256KB）；随着使用次数的增多，文件大小逐渐增长。

```
#qemu-img create -f qcow2 linuxppc.qcow2 200G
```

5．qemu：面向完整 PC 系统的开源仿真器

作用：qemu 是一种支持多种 CPU 的机器模拟器，因此可以在 X86 计算机上运行 PowerPC 软件，在 SPARC 计算机上运行 X86-64 软件，或者其他任意组合。

用法：qemu [选项] [disk_image]

一般选项如下。

– -M machine：选择模拟的机器。
– -fda/-fdb file：使用 file 作为软盘映象。也可以通过将/dev/fd0 作为文件名来使用主机软盘。

- -hda/-hdb file：使用 file 作为硬盘 0,1,2,3 的映象。
- -cdrom file：使用文件作为 CD-ROM 映象（但是不可以同时使用-hdc 和-cdrom）。可以通过使用/dev/cdrom 作为文件名来使用主机的 CD-ROM。
- -boot [a|c|d]：由软盘（a）、硬盘（c）或 CD-ROM（d）启动，在默认情况下由硬盘启动。
- -snapshot：写入临时文件而不写入磁盘映象文件。在这种情况下，没有写回我们所使用的磁盘映象文件。然而我们却可以通过按下 C-a s 来强制写回磁盘映象文件。
- -m megs：设置虚拟内存尺寸为 megs M 字节。在默认情况下为 128M 字节。
- -smp n：模拟一个有 *n* 个 CPU 的 SMP 系统。以 PC 为目标，最多可以支持 255 个 CPU。
- -nographic：在通常情况下，Qemu 使用 SDL 来显示 VGA 输出。使用这个选项，我们可以禁止所有的图形输出，这样 Qemu 只是一个简单的命令行程序。模拟的串口将会重定向到命令行。所以，我们仍然可以在 Qemu 平台上使用串口命令来调试 Linux 内核。
- -k language：使用键盘布局语言（例如 fr 为法语）。这个选项只有在不易得到 PC 键盘的情况下使用。在 PC/Linux 或 PC/Windows 主机上不需要使用这个选项。
- -audio-help：这个选项将会显示声音子系统的帮助、驱动列表及可调用的参数。
- -localtime：设置时钟为本地时间（默认为 UTC 时间）。如果是在 MS-DOS 或 Windows 上，这个选项则需要正确的日期。
- -full-screen：以全屏方式启动。
- -pidfile file：在 file 文件中存放 Qemu 的进程 PID。如果是由脚本启动的，这个选项是相当有用的。
- -win2k-hack：当安装 Windows 2000 时可以使用这个选项来避免磁盘错误。

USB 选项如下。

- -usb：允许 USB 驱动。
- -usbdevice devname：添加 USB 设备名。可以使用查看监视器命令 usb_add 来得到更为详细的信息。

网络选项如下。

- -net nic[,vlan=n] [,macaddr=addr]：创建一个新的网卡并与 VLAN *n*（在默认情况下 *n*=0）进行连接。在 PC 上，NIC 当前为 NE2000。作为可选项的项目，MAC 地址可以进行改变。如果没有指定-net 选项，则会创建一个单一的 NIC。
- -net user[,vlan=n]：使用用户模式网络堆栈，这样就不需要管理员权限来运行。如果没有指定-net 选项，这将是默认的情况。
- -net tap[,vlan=n] [,fd=h] [,ifname=name] [,script=file]：将 TAP 网络接口 name 与 VLAN *n* 进行连接，并使用网络配置脚本 file 进行配置。默认的网络配置脚本为/etc/qemu- ifup。如果没有指定 name，OS 将会自动指定一个。fd=h 可以用来指定一个已经打开的 TAP 主机接口的句柄。
- -net socket[,vlan=n] [,fd=h] [,listen=[host]:port] [,connect=host:port]：使用 TCP socket 将 VLAN *n* 与远程的另一台 Qemu 虚拟机的 VLAN 进行连接。如果指定了 listen，Qemu 将在 port 端口监听连入请求（host 是可选的），connect 可以用来使用 listen 选项与另一个 Qemu 实例进行连接。fd=h 指定了一个已经打开的 TCP socket。

- -net socket[,vlan=n] [,fd=h] [,mcast=maddr:port]：创建一个 VLAN *n*，并使用 UDP 多址通信套接口与其他的 Qemu 虚拟机进行共享，尤其是对于每一台使用多址通信地址和端口的 Qemu 使用同一根总线。
- -net none：表明没有网络设备需要进行配置。如果没有指定-net 选项，则会用来覆盖活跃的默认配置。
- -tftp prefix：当使用用户模式网络堆栈时，激活一个内置的 TFTP 服务器。所有以 prefix 开始的文件将会使用一个 TFTP 客户端从主机下载到本地。在本地的 TFTP 客户端必须以二进制模式进行配置（使用 UNIX 的 TFTP 客户端的 bin 命令）。在客户机上的主机 IP 地址如通常的 10.0.2.2。
- -smb dir-redir [tcp|udp]:host-port:[guest-host]:guest-port：当使用用户模式网络堆栈时，将连接到主机端口 host-port 的 TCP 或 UDP 连接重定向到客户机端口 guest-port 上。如果没有指定客户机端口，则它的值为 10.0.2.15（由内建的 DHCP 服务器指定默认地址）。

Linux 启动选项如下。当我们使用这些选项时，可以使用一个指定的内核，而没有将它安装在磁盘映象中。这对于简单地测试各种内核是相当有用的。

- `-kernel bzImage'：使用 bzImage 作为内核映象。
- `-append cmdline'：使用 cmdline 作为内核的命令行。
- `-initrd file'：使用 file 作为初始的 RAM 磁盘。

调试选项如下。

- `-serial dev'：重定向虚拟串到主机的设备 dev。可用的设备如下。
  - Ø vc：虚拟终端。
  - Ø pty：（Linux）伪 TTY（自动分配一个新的 TTY）。
  - Ø null：空设备。
- /dev/XXX''：（Linux）使用主机的 tty。例如，'/dev/ttyS0'。主机的串口参数通过模拟进行设置。
- /dev/parportN：（Linux）使用主机的并口 N。当前只可以使用 SPP 的并口特征。
- file filename：将输出写入到文件 filename 中。没有字符可读。
- stdio：（UNIX）标准输入/输出。
- pipe filename：（UNIX）有名管道 filename。
- '-parallel dev'：重定向虚拟并口到主机的设备 dev（与串口相同的设备）。在 Linux 主机上，`/dev/parportN'可以被用来使用与相应的并口相连的硬件设备。这个选项可以使用多次，最多可以模拟 3 个并口。
- `-monitor dev'：重定向监视器到主机的设备 dev（与串口相同的设备）。在图形模式下的默认设备为 vc，而在非图形模式下为 stdio。
- '-s'：等待 gdb 连接到端口 1234。
- `-p port'：改变 gdb 连接端口。
- `-S'：在启动时并不启动 CPU（必须在监视器中输入'c'）
- '-d'：输出日志到/tmp/qemu.log。

– `-hdachs c,h,s,[,t]'：强制硬盘 0 的物理参数（1 <= c <= 16383, 1 <= h <= 16, 1 <= s <=63），并且可以选择强制 BIOS 的转换模式（t=none, lba or auto）。通常 Qemu 可以检测这些参数。这个选项对于老的 MS-DOS 磁盘映象是相当有用的。
– `-std-vga'：模拟一个 Bochs VBE 扩展的标准 VGA 显卡（默认情况下为 Cirrus Logic GD5446 PCI VGA）。
– `-loadvm file'：从一个保存状态启动。

在图形模拟时，我们可以使用下面的这些组合键。

– Ctrl+Alt+f：全屏。
– Ctrl+Alt+n：切换虚拟终端'n'。标准的终端映射如下。
  - Ø n=1：目标系统显示。
  - Ø n=2：监视器。
  - Ø n=3：串口。
– Ctrl+Alt：抓取鼠标和键盘。

在虚拟控制台中，我们可以使用 Ctrl+Up、Ctrl+Down、Ctrl+PageUp 和 Ctrl+PageDown 组合键在屏幕中进行移动。

在模拟时，如果我们使用`-nographic'选项，可以使用 Ctrl-a h 来得到终端命令：

– Ctrl-a h：打印帮助信息。
– Ctrl-a x：退出模拟。
– Ctrl-a s：将磁盘信息保存入文件（如果为-snapshot）。
– Ctrl-a b：发出中断。
– Ctrl-a c：在控制台与监视器之间进行切换。
– Ctrl-a a：发送 Ctrl-a。

说明

Qemu 是一个面向完整 PC 系统的开源仿真器。除了仿真处理器外，Qemu 还允许仿真所有必要的子系统，如联网硬件和视频硬件。它还允许实现高级概念上的仿真（如对称多处理系统（多达 255 个 CPU）和其他处理器架构（如 ARM 和 PowerPC）。

应用实例如下。

使用 qemu 命令安装 Windows XP，首先建立磁盘映象。

```
#qemu-img create -f qcow winxp.img 4G
```

然后使用 ISO 文件安装。

```
#qemu -hda winxp.img -cdrom Deepin-LiteXP-SP3.ISO-m 256 -boot d
```

安装开始界面如图 9-13 所示。

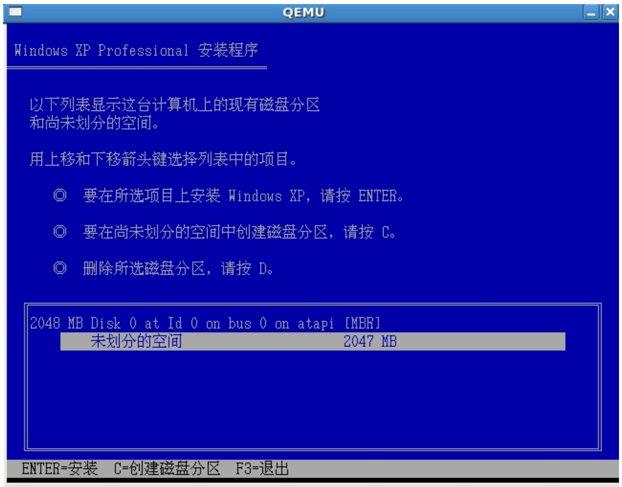


图 9-13　安装开始界面

文件复制完成后使用 qemu 命令从硬盘启动。

```
#qemu -hda winxp.img -m 256 -boot c
```

后面的过程和普通 Windows XP 的安装过程相同。

10 Linux 打印管理命令

10.1 cupsd：通用打印程序守护进程

作用：cupsd 是通用打印程序守护进程。

用法：cupsd [-c config-file] [选项]

主要选项如下。

- -c config-file：设置配置文件，默认配置文件是/etc/cups/cupsd.conf。
- -f：进行前台打印。
- -h：显示帮助信息。
- -F：这个参数让 cupsd 在前台进行，主要用于初始化操作。
- -t：测试配置文件中的语法错误。

应用实例如下。

（1）启动 CUPS 守护进程。

```
# service cups start
启动 cups：                                                    [确定]
```

（2）关闭 CUPS 守护进程。

```
# service cups stop
停止 cups：                                                    [确定]
```

（3）重新启动 CUPS 守护进程。

```
# service cups restart
停止 cups：                                                    [确定]
启动 cups：                                                    [确定]
```

（4）禁用 xerox 打印机功能。

```
# cups disable xerox
```

说明

安装打印机守护进程首先要安装打印机包，这可能是 CUPS，也可能是 LPRng，这些包通常都是与发行版一起提供的。如果需要发行版没有提供的包，那就可能需要寻找一个针对该发行版预先编译的包，或者自己从源代码编译这个包。

CUPS 提供用于设置打印机及使网络中的系统可以访问这些打印机的各种命令。此外，CUPS 还支持多种特定于打印机的选项，通过这些选项可以控制打印机配置。表 10-1 列出了常用的 CUPS 命令。某些 CUPS 命令的名称与传统 LP 打印命令的名称相同，但受 CUPS 管理的命令的行为可能会有所不同。

表 10-1　CUPS 命令行实用程序

命　令	任　务
cancel	取消打印请求
cuspaccept	允许将打印请求排队到指定的目标
cuspdisable	禁用指定的打印机或类
cupsenable	启用指定的打印机或类
cupsreject	拒绝将打印请求排队到指定的目标
lp	提交打印请求
lpadmin	设置或更改打印机或类的配置
lpc	提供对 CUPS 打印队列和类队列的有限控制
lpinfo	显示 CUPS 服务器已知的可用设备或驱动程序
lpmove	将指定作业或所有作业移至新的目标中
lpoptions	显示或设置打印机选项和默认值
lpq	显示当前打印队列状态
lpr	提交打印请求
lprm	取消已排队等候打印的打印作业
lpstat	显示队列和请求的状态信息
cupsctl	配置的 cupsd.conf 选项
cupsfilter	文件转换为另一种格式使用 CUPS 过滤器

10.2 cupsaccept：指示打印系统接受发往指定目标打印机的打印任务

作用：cupsaccept 属于 CUPS 套件，用于指示打印系统接受发往指定目标打印机的打印任务。

用法：cupsaccept [选项]

主要选项如下。

- -E：当连接到服务器时强制加密。
- -U：指定连接服务器时使用的用户名。
- -h：指定连接服务器名和端口号。

应用实例如下。

启用打印机以接受打印请求并打印这些请求。

```
$ cupsaccept printer-name
```

10.3 lpadmin：配置 LP 打印服务

作用：配置 LP 打印服务。

用法：lpadmin [选项]

> **说明**
> 每次合法调用 lpadmin 时都必须只使用-p、-x 或-d 选项中的一个。

主要选项如下。

- -p printer：命名下面的所有 options 所引用的 printer。如果 printer 不存在，则将创建它。
- -x dest：从 LP 系统中删除目标 dest。
- -d[dest]：使现有目标 dest 成为新的系统默认目标。如果没有提供 dest，则没有系统默认目标。当 lpsched 正运行时，可以使用此选项。不允许与-d 选项一起使用。
- -p：指定打印机的名称。此名称最多可由 127 个字符组成。
- -E：使打印机开始打印。如果不提供此选项，则不能使用打印机进行打印。
- -v device：指定打印所用的设备 URL。
- -m：指定已安装的 PPD 文件。
- -c：将打印机加入类。
- -i：为打印机设置“system V”风格的接口脚本。
- -o name=value：为“PPD”或服务器设置选项。
- -r：从类中删除打印机。
- -u：设置打印机用户级的访问控制。
- -D：为打印机提供一个文字描述。
- -E：允许打印机接受打印任务。
- -L：为打印机位置提供一个文字描述。

- -P：为打印机指定一个 PPD 描述文件。
- -h server：主机名称。

带有-p 选项的 lpadmin 命令，可将打印添加到 CUPS 中。

```
lpadmin –p 打印机名称 –E  –V URL 设备 –m PPD 文件
```

说明

lpadmin 命令通过定义打印机和设备来配置 LP 打印服务。它被用于：

- 添加和更改打印机。
- 从服务上删除打印机。
- 设置或更改系统默认目的地。
- 为打印机故障定义警告。
- 安装打印机。
- 为远程打印服务定义打印机。

打印机和类的名称不能长于正在使用的文件系统类型所允许的文件名最大长度，并且可以由除空格、斜杠、反斜杠、冒号、分号、逗号、星号、问号和代字符（～）之外的所有可打印字符构成。除了打印机名的首位，破折号可以用于任何地方。

应用实例如下。

（1）假定现在有一台名为 lp1 的惠普激光打印机，在下面的命令中，它将通过/dev/lp 使用 LaserJet 接口。

```
#lpadmin -plp1 -mlaserjet -v/dev/lp
```

（2）使用 lpadmin 命令。

```
#lpadmin -p LaserJet -E -v parallel:/dev/lp0 -m laserjet.ppd
```

上面的 lpadmin 语法执行以下任务：

- 添加名为 LaserJet 的打印机（-p）。
- 启用要使用的打印机（-E）。
- 设置要使用的设备和设备类型（-v）。
- 使用驱动程序 / PPD 文件 laserjet.ppd（-m）。

（3）删除 xerox1 打印机。

```
# lpadmin -x xerox1
```

（4）添加连接到 Linux 系统并行端口的 HP DeskJet 打印机。

```
#lpadmin –p DesJet –E –v parallel:/dev/lp0 –m deskjet.ppd
```

（5）添加使用 JetDirect 网络接口的 HP LaserJet 打印机，且此接口的地址为 192.168.0.10。

```
   #/usr/sbin/ipadmin －p LaserJet －E －v socket:// 192.168.0.10 －m
laserjet.ppd
```

（6）为 IP 地址是 192.168.0.1 的 xerox1 打印机配置 PostScript 驱动。

```
    #lpadmin -p xerox1 -E -m "postscript.ppd.gz"  -v
"lpd:192.168.0.1/PS-66D975-P1" -D "Xerox 1"
```

（7）添加已连接到并行端口的打印机。

要添加已连接到并行端口的 HP DeskJet 打印机 DeskJet，需要输入以下命令：

```
   $ lpadmin -p DeskJet -E -v parallel:/dev/lp1 -m deskjet.ppd
```

deskjet.ppd 是 CUPS 附带的 HP DeskJet 驱动程序的 PPD 文件。

10.4 lp：打印文件

作用：lp 命令用于将文件打印请求发送到打印机进行打印。

用法：lp [选项]文件

主要选项如下。

- -c：当 lp 命令运行的时候立即复制要打印的文件。lp 命令只有在被请求的时候才复制文件。不创建任何链接。如果指定了-c 标志，注意不要在打印之前删除任一文件。如果没有指定-c 标志，在请求之后对于文件的更改将显示在打印输出上。
- -d Queue：指定将作业发送到的打印队列。
- -m：在文件打印之后发送邮件。默认情况下，在打印请求正常完成的基础上不会发送任何邮件。
- -n Number：指定打印输出的副本数目。副本的默认数目是 1。
- -o Options：指定将特定于后端的标志传递到后端。这样对每个队列而言，在本文中未描述的其他标志可以包含在 lp 命令中（请参阅 piobe 命令获取这些标志的列表）。指定此标志与对 enq 命令指定-o 标志相同。
- -s：禁止自动返回作业数目。lp 命令默认情况下报告作业数目，-s 标志覆盖了默认值。
- -t Title：指定打印输出的标志页上的文件标题。
- -w：文件打印之后在打印请求者终端上写一个消息。

该命令返回以下退出值。

- 0：所有输入文件成功处理。
- >0：没有输出设备可用，或者出现一个错误。

说明

lp 命令安排 Files 参数指定的文件及其关联信息（称为请求）由某个行式打印机打印。如果没有指定 Files 参数值，lp 命令将接受标准输入。文件名——（破折号）代表标准输入，并且除文件之外它能够在命令行被指定。lp 命令以指定顺序发送请求。如果作业被提交到本地打印队列，lp 命令将显示以下标准输出。

```
作业号：nnn
```

其中，nnn 是指定的作业号。要禁止作业号请使用-s 标志。

应用实例如下。

（1）使用文件的一个副本打印/etc/motd 文件的 30 个副本，并且要用邮件通知用户作业完成。

```
#lp  -c  -m  -n30  -dlp0:lpd0 /etc/motd
```

（2）排队 MyFile 文件并返回作业编号。

```
#lp myfile
```

（3）排队 MyFile 文件并禁止作业编号。

```
#lp  -s myfile
```

（4）在连接到设备 dlp0 的打印机 lp0 上打印文件/etc/motd。

```
# lp /etc/motd
```

（5）使用后端标志-f 和-a 并带上作业标题 blah 打印/etc/motd 文件。

```
# lp -t "blah" -o -f -o -a /etc/motd
```

10.5 lpstat：显示行式打印机的状态信息

作用：显示关于行式打印机的当前状态信息。如果没有给出任何标志，lpstat 打印 lp 命令发出的所有请求的状态。

用法：lpstat[选项]

主要选项如下。

- -a：显示打印机的接收状态。
- -c：显示打印类。
- -p：显示打印状态，即启用或禁用。

- -s：显示默认打印机、打印机和类。这个选项等价于-d -c -v。注意，多个选项必须分隔开，因为许多选项都可以指定值。
- -E：与打印机连接时加密。
- -R：显示打印任务的等级。
- -U：指定可选用户名。
- -a：显示接受打印任务的打印机。
- -d：显示默认打印机。
- -h：指定可选的服务器信息。
- -l：显示长格式。
- -t：显示所有的状态信息。

应用实例如下。

（1）显示所有打印队列的状态。

```
#lpstat
```

要显示所有打印机的长状态，请输入如下代码：

```
# lpstat  -t
scheduler is running
no system default destination
device for Cups-PDF: cups-pdf:/
Cups-PDF accepting requests since 2008年09月26日 星期五 04时30分39秒
printer Cups-PDF is idle.  enabled since 2008年09月26日 星期五 04时30分
39秒
```

（2）显示打印队列 lp0 中的一个作业号。

```
#lpstat  -p lp0
```

（3）查看可用打印的列表。

```
# lpstat –p –d
Printer LJ400 is idle
Printer StylusColor600 is idle
System default destination:LJ4000
```

10.6 lpr：排队打印作业

作用：lpr 命令使用假脱机守护进程在设施可用时打印指定的 File 参数。如果没有指定文件，lpr 命令将从标准输入读取。

用法：lpr [选项]

主要选项如下。

- -C,-J 或-T：设置作业名。
- -P：选择特定的打印机。
- -#：指定复制的数目。注意，这与我们在 lp 命令中看到的-n 选项不同。
- -m：在作业完成时发送 E-mail。
- -l：打印文件已为打印进行了格式化。它等价于-o raw。
- -o：设置作业选项。
- -p：使用一个标题对文本文件进行格式化。它等价于-o prettyprint。
- -q：保留作业（或对作业进行排队）以后打印。
- -r：在进行脱机打印之后删除文件。

应用实例如下。

要打印 3 个文件 new.index.c、print.index.c 和 more.c，并指定 3 个副本，请输入如下命令：

```
#cat new.index.c print.index.c more.c | lpr  -#3
```

10.7 lprm：从打印队列中删除任务

作用：lprm 命令从打印机的假脱机队列中删除一个或多个作业。

用法：lprm [-U username] [-h server[:port]] [-P destination[/instance]] [-] [job ID(s)]

主要选项如下。

- -：删除用户拥有的所有作业。root 用户使用该标志可以从排队中删除所有作业。该标志对远程打印无效。
- -P：删除默认打印机的当前打印队列，在/etc/printcap 文件中定义。
- -h server[:port]：服务器端口号。
- job ID(s)：用户 ID。
- -U username：用户名称。
- -E：与打印服务器连接时强制使用加密。

应用实例如下。

（1）从默认打印机队列中删除编号为 13 的作业。

```
#lprm 13
```

（2）从打印机队列 lp0 中删除编号为 13 的作业。

```
#lprm -P lp0 13
```

（3）从打印机队列中删除 cjh 用户的作业。

```
#lprm cjh
```

10.8 lpc：控制打印机

作用：lpc 命令提供（包括 BSD 系统）打印机控制。

用法：lpc

说明

lpc 命令用于控制一个或多个打印机的操作。lpc 命令可以用于启动或停止打印机、禁用或启用打印机的假脱机队列、重排队列中的作业顺序或者显示每台打印机的状态，同时显示其假脱机序列和打印机守护进程。如果没有参数，则 lpc 命令会以 lpc> 作为提示符交互地运行。如果给出了参数，lpc 命令将第一个参数解释为一个要执行的 Command；而每个后继的参数都被当作该命令的 Parameter。标准输入可以被重定向，以便 lpc 命令从文件中读取 Command。

应用实例如下。

（1）启动没有参数的 lpc。

```
# lpc
lpc>
```

查看可用命令。

```
lpc> ?
Commands may be abbreviated.  Commands are:
exit    help    quit    status  ?
```

使用 status 子命令。

```
lpc> status
Cups-PDF:
        printer is on device 'cups-pdf' speed -1
        queuing is enabled
        printing is enabled
        no entries
        daemon present
    退出
lpc> exit
#
```

（2）禁用 xerox 打印机功能。

```
# lpc status xerox
xerox:
```

```
        printer is on device 'lpd' speed -1
        queuing is enabled
        printing is enabled
        no entries
        daemon present
```

10.9 lpq：检查假脱机队列

作用：查询提供当前打印的队列。

用法：lpq[选项]

主要选项如下。

- -l：生成长输出格式。
- +[Number]：显示假脱机队列直到其为空。Number 变量以秒表示重新生成显示前的时间。
- -E：强制使用加密方式与服务器连接。
- -P：显示打印机上的打印队列状态。
- -U：自动可选的用户名。
- -a：报告所有打印机的定义任务。
- -h：指定打印服务器信息。
- +：指定显示状态的间隔时间。

应用实例如下。

使用 lpq 命令检查打印队列。

```
#lpq
xerox is not ready
Rank    Owner   Job    File(s)                          Total Size
1st     brendan 14     RobotPlayer.java                 1024 bytes
2nd     ian     16     .bashrc                          1024 bytes
3rd     ian     17     .bashrc                          1024 bytes
#lpq
xerox is not ready
Rank    Owner   Job    File(s)                          Total Size
1st     brendan 14     RobotPlayer.java                 1024 bytes
2nd     ian     16     .bashrc                          1024 bytes
3rd     ian     17     .bashrc                          1024 bytes
$ lpq -P r220
r220 is ready
no entries
```

```
$ lpq -a
Rank    Owner   Job    File(s)                          Total Size
1st     brendan 14     RobotPlayer.java                 1024 bytes
2nd     ian     16     .bashrc                          1024 bytes
3rd     ian     17     .bashrc                          1024 bytes
```

在这个例子中，有 3 个作业 14、16 和 17 都针对打印机 xerox 在排队。注意，在包含-P 选项时，输出结果显示尚没有打印机就绪。还要注意，用户 ian 两次提交了一个作业，当一个作业第一次没有成功打印时，用户经常会这样操作。可以通过使用 lprm 命令将某个作业从打印队列中删除来防止额外打印一份副本。常见的用户授权设置允许用户删除自己的作业，但是不能删除其他用户的作业。不使用选项时，就会删除当前的打印作业。使用“-”选项，会将所有作业全部删除。

10.10 lpinfo：显示驱动和设备

作用：lpinfo 命令用来显示驱动和设备。

用法：lpinfo [-E] [-l] [-m] [-v]

主要选项如下。

- -E：当连接到服务器时强制使用密码。
- -l：使用长格式输出设备驱动列表。
- -v：在系统上显示所支持的设备列表。
- -m：在系统上显示可供使用的设备驱动。

应用实例如下。

（1）获取受支持设备的完整列表。

```
# lpinfo -v
network socket
network beh
network bluetooth
file cups-pdf:/
direct hal
direct hp:/no_device_found
network http
network ipp
network lpd
direct parallel:/dev/lp0
direct hpfax:/no_device_found
direct canon:/dev/lp0
```

```
direct epson:/dev/lp0
direct scsi
serial serial:/dev/ttyS0?baud=105200
network smb
```

（2）显示 Xerox 可用的打印机驱动程序。

```
# lpinfo -m | grep -i "docuprint.c"
foomatic:Xerox-DocuPrint_C6-cdj550.ppd Xerox DocuPrint C6 Foomatic/cdj550 (recommended)
foomatic:Xerox-DocuPrint_C8-cdj550.ppd Xerox DocuPrint C8 Foomatic/cdj550 (recommended)
foomatic:Xerox-DocuPrint_C10-cdj500.ppd Xerox DocuPrint C10 Foomatic/cdj500
foomatic:Xerox-DocuPrint_C10-hpdj.ppd Xerox DocuPrint C10 Foomatic/hpdj
foomatic:Xerox-DocuPrint_C10-pcl3.ppd Xerox DocuPrint C10 Foomatic/pcl3 (recommended)
foomatic:Xerox-DocuPrint_C20-cljet5.ppd Xerox DocuPrint C20 Foomatic/cljet5
foomatic:Xerox-DocuPrint_C20-gimp-print-ijs.ppd Xerox DocuPrint C20 Foomatic/gimp- print-ijs
foomatic:Xerox-DocuPrint_C20-hpijs.ppd Xerox DocuPrint C20 Foomatic/hpijs
foomatic:Xerox-DocuPrint_C20-Postscript.ppd Xerox DocuPrint C20 Foomatic/Postscript (recommended)
foomatic:Xerox-DocuPrint_C55-Postscript.ppd Xerox DocuPrint C55 Foomatic/Postscript (recommended)
```

说明

此处给出了 Docuprint C20 的几个选择。推荐的驱动程序是 PostScript，因为该打印机可以支持 PostScript 驱动程序。

10.11 lpmove：将作业从一个队列移动到另一个队列

作用：lpmove 命令允许将作业从一个队列移动到另一个队列中。

用法：lpmove [-E] [-h server[:port]] [-U username] job destination

主要选项如下。

- -E：队列移动时强制使用密码。
- -U username：指定用户名。
- -h server[:port]：指定服务器及其端口（默认是 631）。
- Job：作业。
- Id：作业 ID。

应用实例如下。

如果打印机 xerox 现在没有运行，我们就需要使用这种功能。lpmove 命令需要一个 root 授权用户。

```
# lpmove xerox-16 anyprint
# lpmove xerox r220
# lpq
xerox is not ready
no entries
# lpq -a
Rank Owner Job File(s) Total Size
active ian 18 fig1.gif 26624 bytes
```

上面的例子显示了如何将这些作业移动到另一个队列上，先按打印机和作业 ID 来指定，然后指定某个打印机的所有作业。当再次查看这个队列时，两个作业都已经打印完毕。

10.12 cancel：取消已存在的打印任务

作用：cancel 命令用于取消已存在的打印任务。

用法：cancel（选项）

主要选项如下。

- -a：取消所有打印任务。
- -E：当连接到服务器时强制使用加密。
- -U：指定连接服务器时使用的用户名。
- -u：指定打印任务所属的用户。
- -h：指定连接的服务器名和端口号。

10.13 cupsdisable：禁用指定的打印机或类

作用：cupsdisable 命令属于 CUPS 套件，用于停止指定的打印机或类。

用法：cupsdisable [选项]

主要选项如下。

- -E：当连接到服务器时强制使用加密。
- -U：指定连接服务器时使用的用户名。
- -u：指定打印任务所属的用户。
- -c：取消指定打印机的所有打印任务。
- -h：指定连接的服务器名和端口号。
- -r：指定停止打印机的原因。

应用实例如下。

停止打印请求。

```
$ cupsdisable printer-name
```

10.14 cupsreject：指示打印系统拒绝发往指定目标打印机的打印任务

作用：cupsreject 命令属于 CUPS 套件，用于指示打印系统拒绝发往指定目标打印机的打印任务。

用法：cupsreject [选项]

主要选项如下。

- -E：当连接到服务器时强制使用加密。
- -U：指定连接服务器时使用的用户名。
- -h：指定连接的服务器名和端口号。
- -r：指定拒绝打印任务的原因。

10.15 cupsenable：启动指定的打印机

作用：cupsenable 命令用于启动指定的打印机。

用法：cupsenable[选项]

主要选项如下。

- -E：当连接到服务器时强制使用加密。
- -U：指定连接服务器时使用的用户名。
- -u：指定打印任务所属的用户。
- -h：指定连接的服务器名和端口号。

11 Linux 库应用命令

11.1 ldconfig：配置查找共享库

作用：ldconfig 命令执行两个基本函数处理共享库。首先创建连接，这样才能保证到共享库的引用总是最新版本。然后在文件/etc/ld.so.cache 中保存有效共享库的完整列表。

用法：ldconfig [选项]

主要选项如下。

- -c,--format=FORMAT：采用的格式——新、旧或兼容（默认）。
- -C CACHE：将 CACHE 用作缓冲区文件。
- -f CONF：将 CONF 用作配置文件。
- -i,--ignore-aux-cache：忽略辅助缓存文件。
- -l：手工连接独立的库。
- -n：只在命令行中给出了进程目录。未创建缓冲区。
- -N：不要创建缓冲区。
- -p,--print-cache：打印缓冲区。
- -r ROOT：进入 ROOT 目录并将其作为根目录。
- -v,--verbose：生成详细消息。
- -X：不要生成连接。
- -?,--help：给出该系统求助列表。
- --usage：给出简要的用法信息。
- -V,--version：打印程序版本号。

长选项的强制或可选参数对相应的短选项也是强制或可选的。

应用实例如下。

1）在系统中查找动态链接库

有时候用户想知道系统中有哪些动态链接库，或者想知道系统中有没有某个动态链接库，这时，可用-p 选项让 ldconfig 输出缓存文件中的动态链接库列表，从而查询得到。

查看 MySQL 相关的动态链接库。

```
# ldconfig -p |grep mysql
  libmysqlclient.so.18 (libc6,x86-64) => /usr/lib64/mysql/libmysqlclient.so.18
```

2）创建所有必要的新连接，生成文件/etc/ld.so.cache 的新版本

将 usr/local/lib 加入动态链接库。

```
# ldconfig /usr/local/lib
```

说明

工具 ldconfig 执行两个基本函数处理共享库。首先创建连接，这样才能保证到共享库的引用总是最新版本。然后在文件/etc/ld.so.cache 中保存有效共享库的完整列表。工具 ldconfig 读出文件/etc/ld.so.conf，它是包含共享库的目录列表，然后使用这些目录名（还有目录/lib 和/usr/lib）来定位库，这些库用于连接，列在目录/etc/ld.so.cache 中。文件/etc/ld.so.conf 中的目录名可由新的一行、分号、制表符或空格分隔。/etc/ld.so.cache 的内容不是文本用法，也不能被编译。在构造/etc/ld.so.cache 文件之前，ldconfig 会分析库的名字和内容，并创建动态连接，这样就会载入最新版本的库。例如，载入 libdl.so.2 的程序通过连接，可能实际载入的是 libdl-2.2.4.so。当公布了修复问题的新版本库（例如，libdl-2.2.5.so 或 libdl-2.3.0.so）后，工具 ldconfig 就会更新连接 libld.so.2 指向新版本。但如果新版本 libdl-3.0.0.so 可能打破旧程序，那么旧连接就不会受到干扰，而会创建新的连接 libdl.so.3。这种命名约定使得可能在同一环境下运行共享库的新、旧两个版本。因为可能需要访问外设，有必要以 root 登录运行 ldconfig。下面的命令创建所有必要的新连接，生成文件/etc/ld.so.cache 的新版本：

```
#ldconfig-v
```

选项-v 生成所有连接的列表，其中包括处理过程中的其他信息。

11.2 nm：列举目标文件中的符号名

作用：工具 nm 用来列出目标文件、静态文档库或动态库中定义的（或是引用此处的）符号。

用法：nm[选项]

主要选项如下。

- -A：同选项--print-file-name。
- -a：同选项--debug-syms。
- -B：同选项--format=bsd。这是默认设置。
- -C 或--demangle：将低级符号名解码（Demangle）成用户级名字，这样可以使得 C++函数名具有可读性。
- -D：同选项--dynamic。

- --debug-syms：显示调试器使用的符号。通常不会显示这些符号。
- --demangle[=type]：demangles 符号，使它变回源代码中找到的用户级的名字。如果指定类型，则会为 auto、gnu、lucid、arm、hp、edg、gnu-v3、java、gnat 或 compaq 类型之一。
- --dynamic：对于动态目标，例如共享库，该选项可以显示动态符号而不是普通符号。
- --extern-only：只显示定义为外部的符号。
- -ffmt：同选项--format。
- --format=fmt：使用指定的输出格式显示符号。可选的格式包括 bsd、sysv 和 posix，其中 bsd 为默认格式。
- -g：同选项--extern-only。
- --help：显示选项列表，然后退出。
- -l：同选项--line-numbers。
- --line-numbers：使用文件中保存的调试信息来确定文件名和每个符号的行号。
- -n：同选项--numeric-sort。
- --no-sort：指出不要将符号排序。
- --numeric-sort：按照符号地址的数值排序。
- -o：同选项--print-file-name。
- -p：同选项--no-sort。
- -P：同选项--format=posix。
- --portability：同选项--format=posix。
- --print-armap：在列举静态库成员符号时，包含模块的索引信息及其他信息。索引的内容包含模块对应的名字映射。
- --print-file-name：用源文件的名字标记每个符号，而不是只在文件头命名源文件。
- -r：同选项--reverse-sort。
- --radix=base：指出打印符号值的数字的进制。可选的为 d 的十进制、o 的八进制或 x 的十六进制。
- --reverse-sort：反序排列，不论字母还是数字均可。
- -s：同选项--print-armap。
- --size-sort：按照尺寸进行符号排序。计算尺寸取的是下一个符号的最高地址和本符号的地址的差。输出列出的是尺寸而不是通常的地址。
- -tbase：同选项--radix。
- --target=bfdnamebfdname：是目标文件用法的名字，它不是当前机器的名字。
- -u：同选项--undefined-only。
- --undefined-only：只显示文件引用但未定义的符号。
- -V：同选项--version。
- --version：显示版本信息，然后退出。

输出符号类型说明如下。

- A：该符号的值是绝对的，在以后的链接过程中，不允许进行改变。这样的符号值常常

出现在中断向量表中，例如用符号来表示各个中断向量函数在中断向量表中的位置。

- B：该符号的值出现在非初始化数据段（bss）中。例如，在一个文件中定义全局 static int test，则该符号 test 的类型为 b，位于 bss section 中，其值表示该符号在 bss 段中的偏移。一般而言，bss 段分配于 RAM 中。
- C：该符号为 common。common symbol 是未初始化数据段。该符号没有包含于一个普通 section 中。只有在链接过程中才进行分配。符号的值表示该符号需要的字节数。例如在一个 c 文件中定义 int test，并且该符号在其他地方会被引用，则该符号类型即为 C；否则其类型为 B。
- D：该符号位于初始化数据段中，一般来说，其分配到 data section 中。例如定义全局 int baud_table[5] = {9600, 19200, 38400, 57600, 115200}，则会分配于初始化数据段中。
- G：该符号也位于初始化数据段中，主要用于 small object 提供访问 small data object 的一种方式。
- I：该符号是对另一个符号的间接引用。
- N：该符号是一个 debugging 符号。
- R：该符号位于只读数据区。
- S：该符号位于非初始化数据区，用于 small object。
- T：该符号位于代码区 text section。
- U：该符号在当前文件中是未定义的，即该符号的定义在其他文件中。例如，当前文件调用另一个文件中定义的函数，这个被调用的函数在当前文件中就是未定义的，但是在定义它的文件中其符号类型是 T。对于全局变量来说，在定义它的文件中，其符号类型为 C；在使用它的文件中，其符号类型为 U。
- V：该符号是一个 weak object。
- W：该符号是 a.out 格式文件中的 stabs symbol。
- ?：该符号类型没有定义。

应用实例如下。

（1）列出目标模块的名字，以及库 libc.a 定义或引用的所有符号。

```
$nm libc.a
```

（2）查找“cos”函数。

如果知道一个函数的名字，但是不知道这个函数是在什么库中定义的，那么可以使用 mn 的“-o”选项和 grep 命令来查找该库的名字。“-o”选项使得显示的每一行都有这个函数库文件名。例如，若要查找“cos”是在什么地方定义的，可以使用下面的命令：

```
#nm -o /lib/* /usr/lib/* /usr/lib/*/* /usr/local/lib/* 2> /dev/null
```

如果 nm 命令没有指定文件名，则会假设使用 a.out。利用命令行选项，可以根据符号的地址、尺寸或名字组织这些符号，而且可以按照很多方式格式化该输出结果。符号也可以被解码（demangled），产生的结果和源代码中的一样。

11.3 ar：建立修改文件或从文件中抽取成员

作用：ar 命令用来建立、修改、提取归档文件。

用法：ar [选项] archive files...

主要选项如下。

- d：从归档文件中删除文件。
- m[ab]：在归档文件中移动文件。
- p：打印在归档文件中找到的文件。
- q[f]：将文件快速追加到归档文件中。
- r[ab] [f] [u]：替换归档文件中已有的文件或加入新文件。
- s：即使没有改动文档，也会创建新的文档索引。这和使用 ranlib 命令的结果一样。
- t：显示归档文件的内容。
- x[o]：从归档文件中分解文件。

特定命令修饰符如下。

- [a]：将文件置于[成员名]之后。
- [b]：将文件置于[成员名]之前（与[i]相同）。
- [P]：在匹配时使用完整的路径名。
- [o]：保留原来的日期。
- [u]：只替换比当前归档内容更新的文件。

通用修饰符如下。

- [c]：不在必须创建库的时候给出警告。
- [s]：创建归档索引（cf.ranlib）。
- [S]：不要创建符号表。
- [T]：makeathinarchive。
- [v]：输出较多信息。
- [V]：显示版本号。

说明

ar 命令是很老的 UNIX 工具之一，它的语法和其他一些老工具很类似，例如 tar。这些工具首先出现的是选项标签，而选项的字母表示之间没有空格，而且选项可通过（也可不通过）连字符来表达。可选的命令行项目 positionname 和 count 只有当用到了某些需要它们的选项时才会出现。ar 命令的选项分为两类：命令行选项告诉 ar 应该采取什么样的操作（有效命令行选项中的一个）；修改（modifier）子选项指出如何执行该命令操作。ar 支持的 CPU 类型：elf64-x86-64 elf32-i386 elf32-x86-64 a.out-i386-linux pei-i386 pei-x86-64 elf64-l1om elf64-k1om elf64-little elf64-big elf32-little elf32-big plugin srec symbolsrec verilog tekhex binary ihex。

应用实例如下。

创建一个静态库，或将目标代码加入到已经存在的静态库中。

```
# ar rcs my_libraty.a file1.o file2.o
```

以上命令表示要把目标代码 file1.o 和 file2.o 加入到静态库 my_library.a 中。若 my_library.a 不存在，则会自动创建。静态库创建成功后，需要连接到应用程序中来使用。如果使用 gcc 来产生执行程序，需要利用-l 选项来指定静态库（更多信息请查看 gcc 使用手册）。在使用 gcc 时，要注意其参数的顺序。-l 是连接器选项，一定要放在被编译的文件名称之后；若放在文件名称之前，则会连接失败，并会出现莫名其妙的错误。也可以直接使用 d 命令，使用其选项-l 或-L。但最好使用 gcc，因为 ld 的接口有可能会发生变化。

11.4　strip：去除目标文件中的无用信息

作用：strip 命令可以去除目标文件或命令指定文件的调试符号表信息。该目标文件可以是静态库、共享库或者编译程序产生的.o 文件。该命令依赖于文件中包含了多少调试信息，剥离命令（stripping）可动态地削减文件的大小。

用法：strip [选项]

主要选项如下。

- --discard-all：去掉所有非全局符号。
- --discard-locals：去掉编译程序生成的局部符号。通常由字母 L 或句号（.）开头。
- -F bfdname：同选项--target。
- -g：同选项--strip-debug。
- -h：显示项目列表，然后退出。
- -Ibfdname：同选项--input-target。
- --I nput-target=bfdname：将输入目标文件看作指定 bfdname 用法的文件。
- -Kname：同选项--keep-symbol。
- --keep-symbol=name：只将指定符号复制到输出文件中。保存多个名字要多次使用该选项。
- -Nname：同选项--strip-symbol。
- -Obfdname：同选项--output-target。
- -ofilename：不是重写原始文件，输出被写到新指定的文件中。该选项会限制命令，只能操作一个文件。
- --output-target=bfdname：取代原始文件为剥离（stripped）后的文件，用法满足 bfdname。
- -p：同选项--preserve-dates。
- --preserve-dates：最新剥离后的文件和原始输入文件一样，具有相同的访问次数。
- -Rname：同选项--remove-section。

- --remove-section=name：从目标文件中去除指定的章节。本选项可多次使用以除去多个章节。
- -s：同选项--strip-all。
- -S：同选项--strip-debug。
- --strip-all：去掉所有符号，包括连接必要的重定位信息。
- --strip-debug：只去掉调试所必需的符号。
- --strip-symbol=name：去掉指定符号。该选项可多次使用，也可与其他剥离选项一起使用。
- --strip-unneeded：去掉对重定位代码没有必要的所有符号。
- --target=bfdname：将输入用法和输出用法都设置为指定的 bfdname。
- -v：同选项--verbose。
- --verbose：通过列举所有剥离后的文件，产生更详细的输出。
- -x：同选项--discard-all。
- -X：同选项--discard-locals。

应用实例如下。

下面的命令会剥离目标文件 main.o 及 libglom.a 库中所有目标文件的所有调试信息。

```
$stripmain.olibglom.a
```

工具 strip 将现有文件替换为剥离后的版本，因此如果想要保留原始未剥离的版本，需要在剥离之前先保存副本，或者使用-o 选项生成不同的文件名。

11.5 objdump：展开目标文件、静态库和共享库中的信息

作用：objdump 用于展开目标文件、静态库和共享库中的信息，然后将其表示成可读模式。该命令可用于输出（dump）来自不同用法目标文件的信息。

用法：objdump [选项]

执行 objdump 命令从文件中展开信息，必须利用表 11-1 中的一个或多个选项（每个都有短模式和长模式）指出需要展开的是什么信息。表 11-2 列出的附加选项用来精简输入数据的选择及格式化输出数据。

表 11-1　objdump 选项的短用法和长用法

短　选　项	长　选　项	功　　能
-a	--archive-headers	文档头文件信息
-d	-disassemble	可执行代码的汇编语言
-D	--disassemble-all	可执行代码和数据的汇编语言
-f	--file-headers	完整文件头的内容
-g	--debugging	调试信息

续表

短　选　项	长　选　项	功　　能
-G	--stabs	所有 STABS 信息的原始形式（rawform）
-h	--section-headers	章节头的内容
-i	--info	支持的目标用法和结构列表
-p	--private-headers	指定目标用法的文件头内容
-r	--reloc	重定位信息
-R	--dynamic-reloc	动态重定位信息
-S	--source	混合源代码的可执行汇编语言
-s	--full-contents	混合源代码的所有代码的汇编语言
-t	--syms	符号表内容
-T	-dynamic-syms	动态符号表内容
-x	--all-headers	所有头内容
-V	--version	版本信息

表 11-2　objdump 修改子的命令行选项

命令行选项	含　义
--adjust-vma=offset	将指定位移 offset 添加到所有显示章节的地址中
--architecture=machine	指定反汇编目标文件时使用的架构，当待反汇编文件本身没描述架构信息的时候(比如 S-records)，这个选项很有用。可以用-i 选项列出这里能够指定的架构
--demangle=type	假设符号为指定类型且经过 demangled，则变回源代码中的样式。有效的类型为 auto、gnu、lucid、arm、hp、edg、gnu-v3、java、gnat 和 compaq
--disassembler-options=op	指定 op 为一个或多个传递给分解器（Disassembler）的选项
--disassemble-zeroes	指出分解代码时不要跳过零块
--endian=which	指出输入目标文件为 big-endian 或 little-endian。这里的 little 是指 littleendian，而 big 是指 bigendian
--file-start-context	使用-S 选项时，该选项会从文件一开始就包括上下文信息
--line-numbers	指出输入的行号和文件名
--prefix-addresses	打印每个分解的指令附带的完整地址信息
--section=name	限制目标文件指定章节的信息显示
--show-raw-insn	显示十六进制的操作代码（opcodes）和记忆的汇编语言指令
--targetbfdname	指出输入目标文件的用法。要确定用法是否有效，可输入命令 objdump－i
--wide	格式化超过 80 列的输出
--start-address=address	只处理地址大于指定地址值的数据
--stop-address=address	只处理地址小于指定地址值的数据

应用实例如下。

要列出目标文件 helloworld.o 的头文件和章节头，需要假设输入代码为 bigendian，输入如下命令：

```
$objdump –f –h --endian=big  helloworld.o
```

11.6 ldd：显示共享库的依赖情况

作用：ldd 显示每个程序需要的共享库（Shared Library），程序名在命令行上给出。

用法：ldd[-vVdr]程序

主要选项如下。

- -v：显示 ldd 的版本号。
- -V：显示动态连接器 ld.so 的版本号。
- -d：进行重定位（Relocation），而且报告缺少的函数（仅限于 ELF）。
- -r：对数据目标（Dataobject）和函数进行重定位，而且报告缺少的数据目标（仅限于 ELF）。

说明

对于 a.out 程序，ldd 简单地进行 fork 和 exec 各个程序，执行程序时 argc 参数等于零，a.out 的动态连接器（Dynamic Linker）ld.so 正常情况下能够调入共享库，注意到这个特殊情况，从而显示出依赖关系。对于 ELF 程序，ldd 设置适当的环境变量集，然后 fork 和 exec 各个程序。ELF 的动态连接器 ld-linux.so 正常情况下能够调入共享库，注意到这个特殊情况，从而显示出依赖关系。

应用实例如下。

（1）查询共享库。

PHP 语言是笔者喜欢使用的一种“脚本语言”，如果需要查询哪些共享库，可以首先使用 find 命令查询这个程序的绝对路径，然后使用如下命令：

```
#find –name php
# ldd  /usr/bin/php
       linux-gate.so.1 =>  (0x008e3000)
       libcrypt.so.1 => /lib/libcrypt.so.1 (0x02ce5000)
       libaspell.so.15 => /usr/lib/libaspell.so.15 (0x00549000)
       libpspell.so.15 => /usr/lib/libpspell.so.15 (0x00441000)
       libgmp.so.3 => /usr/lib/sse2/libgmp.so.3 (0x00a2e000)
       libdb-4.3.so => /lib/libdb-4.3.so (0x02e9d000)
```

```
        libpthread.so.0 => /lib/i686/nosegneg/libpthread.so.0 (0x00413000)
        libcurl.so.3 => /usr/lib/libcurl.so.3 (0x0050b000)
        libbz2.so.1 => /usr/lib/libbz2.so.1 (0x02d15000)
        libz.so.1 => /usr/lib/libz.so.1 (0x0042c000)
        libpcre.so.0 => /lib/libpcre.so.0 (0x004ec000)
        libresolv.so.2 => /lib/libresolv.so.2 (0x00a19000)
        libm.so.6 => /lib/i686/nosegneg/libm.so.6 (0x003e4000)
        libdl.so.2 => /lib/libdl.so.2 (0x0040d000)
        libnsl.so.1 => /lib/libnsl.so.1 (0x006a0000)
        libxml2.so.2 => /usr/lib/libxml2.so.2 (0x00110000)
        libgssapi_krb5.so.2 => /usr/lib/libgssapi_krb5.so.2 (0x00b38000)
        libkrb5.so.3 => /usr/lib/libkrb5.so.3 (0x00bfa000)
        libk5crypto.so.3 => /usr/lib/libk5crypto.so.3 (0x00b8a000)
        libcom_err.so.2 => /lib/libcom_err.so.2 (0x00a14000)
        libssl.so.6 => /lib/libssl.so.6 (0x002a0000)
        libcrypto.so.6 => /lib/libcrypto.so.6 (0x007a6000)
        libidn.so.11 => /usr/lib/libidn.so.11 (0x00475000)
        libc.so.6 => /lib/i686/nosegneg/libc.so.6 (0x00c82000)
        libstdc++.so.6 => /usr/lib/libstdc++.so.6 (0x002e5000)
        libgcc_s.so.1 => /lib/libgcc_s.so.1 (0x00242000)
        /lib/ld-linux.so.2 (0x00285000)
        libkrb5support.so.0 => /usr/lib/libkrb5support.so.0 (0x0024e000)
```

输出说明：箭头左边的一列显示的是 PHP 语言所需的.so 文件名，箭头右边的一列显示的是库的真实库名称。应用程序链接库的 so 名字是到实际库的符号链接。以第二行为例，最后的 0x02ce5000 是库 libcrypt.so.1 的加载地址。可以看到运行 PHP 语言需要 28 个共享库。

（2）列出 Linux 系统上 bash shell 程序使用的共享库。

```
# ldd /bin/bash
        linux-gate.so.1 =>  (0x00b7d000)
        libtermcap.so.2 => /lib/libtermcap.so.2 (0x0042c000)
        libdl.so.2 => /lib/libdl.so.2 (0x0040d000)
        libc.so.6 => /lib/i686/nosegneg/libc.so.6 (0x002a2000)
        /lib/ld-linux.so.2 (0x00285000)
```

11.7 patch 命令：修补文件

作用：patch 命令让用户利用设置修补文件的方式，修改、更新原始文件。

用法：patch [选项]...[originalfile] [patchfile]

主要选项如下。

- -b 或--backup：备份每个原始文件。
- -c 或--context：把修补数据解译成关联性的差异。
- -d<工作目录>或--directory=<工作目录>：设置工作目录。
- -D<标识符号>或--ifdef=<标识符号>：用指定的符号把改变的地方标识出来。
- -e 或--ed：把修补数据解译成 ed 指令可用的叙述文件。
- -E 或--remove-empty-files：若修补过后输出的文件其内容是一片空白，则移除该文件。
- -f 或--force：此参数的效果和指定“-t”参数类似，但会假设修补数据的版本为新版本。
- -F<监别列数>或--fuzz<监别列数>：设置最大模糊因子。此标志只应用于上下文差异列表，它使 patch 命令在确定块（hunk）的安装位置时忽略指定行编号。如果没有指定 -F 标志，默认模糊因子为 2。此因子不可以设置成大于上下文差异列表中内容的行的数目（通常为 3）。
- -g<控制数值>或--get=<控制数值>：设置以 RSC 或 SCCS 控制修补作业。
- -i<修补文件>或--input=<修补文件>：读取指定的修补文件。
- -l 或--ignore-whitespace：忽略修补数据与输入数据的跳格、空格字符。
- -n 或--normal：把修补数据解译成一般性的差异。
- -N 或--forward：修补的数据较原始文件的版本更旧，或该版本的修补数据已使用过。
- -o<输出文件>或--output=<输出文件>：设置输出文件的名称，修补过的文件会以该名称存放。
- -p<剥离层级>或--strip=<剥离层级>：设置欲剥离几层路径名称。
- -f<拒绝文件>或--reject-file=<拒绝文件>：设置保存拒绝修补相关信息的文件名称，默认文件名称为.rej。
- -R 或--reverse：假设修补数据是由新旧文件交换位置而产生的。
- -s 或--quiet 或--silent：不显示指令执行过程，除非发生错误。
- -t 或--batch：自动略过错误，不询问任何问题。
- -T 或--set-time：此参数的效果和指定“-Z”参数类似，但以本地时间为主。
- -u 或--unified：把修补数据解译成一致化的差异。
- -Z 或--set-utc：把修补过的文件更改，存取时间设为 UTC。
- -backup-if-mismatch：修补数据不完全吻合，并且没有指定要备份文件时才备份文件。
- -binary：以二进制模式读/写数据，而不通过标准输出设备。
- -verbose：详细显示指令的执行过程。
- Originalfile：源文件。
- Patchfile：修补文件。

说明

尽管并没有指定 patch 和 diff 的关系，但通常 patch 都使用 diff 的结果来完成打补丁的工作，这和 patch 本身支持多种 diff 输出文件用法有很大关系。patch 通过读入 patch 命令文件（可以从标准输入），对目标文件进行修改。通常先用 diff 命令比较新老版本，patch 命令文件则采用 diff 的输出文件，从而保持原版本与新版本一致。如果 patchfile 为空，则从标准输入读取 patchfile 内容；如果 originalfile 也为空，则从 patchfile（肯定来自标准输入）中读取需要打补丁的文件名。因此，如果需要修改的是目录，则必须在 patchfile 中记录目录下的各个文件名。

应用实例如下。

配合使用 diff 和 patch 升级源码。

假设 program-2.0 目录中为老版本，现开发完成的新版本位于 program-3.0 目录中，将两个目录置于同一父目录下，然后在该父目录上执行如下命令：

```
diff -Nur program-2.0 program-3.0 >program-3.0.patch
```

将生成一个 program-3.0.patch 的补丁文件，压缩后发布该补丁文件。假设生成的是 program-3.0.patch.bz2 文件，则在 program-2.0 目录中执行如下命令：

```
bzcat program-2.0.patch.bz2 | patch -p0
```

如此即完成了从 2.0 到 3.0 的升级。如果希望恢复到原版本，可以使用-R（--reverse）参数，但仅对上下文用法的 diff 文件有效。还有一个备份参数也可以使用，但在简单应用中，整个目录备份可能更方便一些。

11.8　ld：连接器

作用：ld 命令合并一组目标文件（Object）和库文件（Archive），重定位数据部分，构建符号引用（Symbol Reference）。一般说来，编译生成可执行文件的最后步骤就是调用 ld。

用法：ld [选项]文件

主要选项如下。

- -b：指定目标代码输入文件的格式。
- -Bstatic：只使用静态库。
- -Bdynamic：只使用动态库。
- -Bsymbolic：把引用捆绑到共享库中的全局符号。
- -c：从指定的命令文件读取命令。
- --cref：创建跨引用表。

- -d：设置空格给通用符号，即使指定了可重定位输出。
- -defsym：在输出文件中创建指定的全局符号。
- -demangle：在错误消息中还原符号名称。
- -e：使用指定的符号作为程序的初始执行点。
- -E：对于 ELF 格式文件，把所有符号添加到动态符号表。
- -f：对于 ELF 格式共享对象，设置 DT_AUXILIARY 名称。
- -F：对于 ELF 格式共享对象，设置 DT_FILTER 名称。
- -format：指定目标代码输入文件的格式（和-b 相同）。
- -g：被忽略，用于提供和其他工具的兼容性。
- -h：对于 ELF 格式共享对象，设置 DT_SONAME 名称。
- -i：执行增量连接。
- -l：把指定的存档文件添加到要连接的文件清单。
- -L：把指定的路径添加到搜索库的目录清单。
- -M：显示连接映射，用于诊断目的。
- -Map：创建指定的文件来包含连接映射。
- -m：模拟指定的连接器。
- -N：指定读取/写入文本和数据段。
- -n：设置文本段为只读。
- -noinhibit-exec：生成输出文件，即使出现非致命连接错误。
- -no-keep-memory：为内存使用优化连接。
- -no-warn-mismatch：允许连接不匹配的目标文件。
- -O：生成优化了的输出文件。
- -o：指定输出文件的名称。
- -oformat：指定输出文件的二进制格式。
- -R：从指定的文件读取符号名称和地址。
- -r：生成可重定位的输出（称为部分连接）。
- -rpath：把指定的目录添加到运行时库搜索路径。
- -rpath-link：指定搜索运行时共享库的目录。
- -S：忽略来自输出文件的调试器符号信息。
- -s：忽略来自输出文件的所有符号信息。
- -shared：创建共享库。
- -sort-common：在输出文件中不按照长度对符号进行排序。
- -split-by-reloc：按照指定的长度在输出文件中创建额外的段。
- -split-by-file：为每个目标文件在输出文件中创建额外的段。
- --section-start：在输出文件中指定的地址定位指定的段。
- -T：指定命令文件（和-c 相同）。
- -Ttext：使用指定的地址作为文本段的起始点。
- -Tdata：使用指定的地址作为数据段的起始点。

- -Tbss：使用指定的地址作为 bss 段的起始点。
- -t：在处理输入文件时显示它们的名称。
- -u：强制指定符号在输出文件中作为未定义符号。
- -warn-common：当一个通用符号和另一个通用符号结合时发出警告。
- -warn-constructors：如果没有使用任何全局构造器，则发出警告。
- -warn-once：对于每个未定义的符号只发出一次警告。
- -warn-section-align：如果为了对齐而改动了输出段地址，则发出警告。
- --whole-archive：对于指定的存档文件，在存档中包含所有文件。
- -X：删除所有本地临时符号。
- -x：删除所有本地符号。

说明

ld 可以读取用连接器命令语言编写的脚本文件，它能够对连接过程提供精确和全面的控制，ld 能够以多种格式读入、连接和输出目标文件，如 COFF 或 a.out，能够把不同的格式连接在一起，产生各种有效的目标文件。用“objdump-i”可以列出各种体系结构支持的格式列表；GNU linker 不仅灵活强大，还能够比其他 linker 提供更多的诊断信息。很多 linker 只要碰上一个错误就立刻停止执行；而 ld 一有可能仍然继续执行，这样容易定位出其他错误（某些情况下，尽管出了错，仍然生成输出文件）。GNU linker ld 期望实现更广泛的适用范围，尽可能兼容其他 linker。通过命令行和环境变量，用户可以用 ld 执行多种操作。ld 后面跟的可以是不同的链接格式，如果什么都不带应该指的是默认的格式。.o 文件是链接目标。

应用实例如下。

在一个标准的、提供相应支持的 UNIX 系统中，常常用 ld 连接标准的 UNIX 目标文件。如果要连接 hello.o，可执行如下命令。

```
$ ld -o output /lib/crt0.o hello.o –lc
```

它告诉 ld 生成一个叫作 output 的文件，其中连接了文件/lib/crt0.o 和 hello.o，以及库文件 libc.a。

12 Linux 开发应用命令

12.1 as：标准 GNU 汇编程序

作用：as 命令是标准 GNU 汇编程序。as 命令读取和汇编指定的文件（约定情况下，此文件以.s 结束）。如果不指定文件，则 as 命令读取和汇编标准输入。默认情况下，它将其输出存储于扩展名为.o 的文件中。运行 as 命令要安装 binutils 软件包。

用法：as[选项] [asmfile...]

主要选项如下。

- -a：显示列表。
- -ac：忽略失败条件。
- -ad：忽略调试指令。
- -ah：包括高级源。
- -al：包括装配。
- -am：包括宏扩展。
- -an：忽略形式处理。
- -as：包括符号。
- =file：设置列出文件的名字。
- --alternate：以交互宏模式开始。
- -f：跳过空白和注释预处理。
- -D：生成汇编调试信息。
- --execstack：要求可执行的堆栈。
- --noexecstack：不要求可执行的堆栈。
- -g --gen-debug：生成汇编调试信息。
- -I DIR：增加目录到搜索目录。
- -J：程序溢出不报警。
- -L,--keep-locals：保持本地符号信息。
- -o OBJFILE：将汇编进程的输出写至指定文件而非写至 a.out 文件（默认是 a.out）。
- -R：生成目标文件后删除源文件。

– --statistics：打印静态变量信息。
– -W --no-warn：禁止显示所有警告。
– --warn：显示所有警告。
– -march=CPU/-mtune=CPU：根据 CPU 进行代码优化。

Linux 汇编语法格式：

绝大多数 Linux 程序员以前只接触过 DOS/Windows 下的汇编语言，这些汇编代码都是 Intel 风格的。但在 UNIX 和 Linux 系统中，更多采用的还是 AT&T 格式，二者在语法格式上有着很大的不同，如表 12-1 所示。

表 12-1　汇编语法格式对比

汇编对象	AT&T 格式	Intel 格式
寄存器名	在 AT&T 汇编格式中，寄存器名要加上“%”作为前缀	而在 Intel 汇编格式中，寄存器名不需要加前缀
立即操作数	在 AT&T 汇编格式中，用“$”前缀表示一个立即操作数	而在 Intel 汇编格式中，立即数的表示不用带任何前缀
源操作数和目标操作数的位置	在 AT&T 汇编格式中，目标操作数在源操作数的右边	在 Intel 汇编格式中，目标操作数在源操作数的左边
操作数的字长	在 AT&T 汇编格式中，操作数的字长由操作符的最后一个字母决定，后缀“b”、“w”、“l”分别表示操作数为字节（byte，8 比特）、字（word，16 比特）和长字（long，32 比特）	在 Intel 汇编格式中，操作数的字长用“byte ptr”和“word ptr”等前缀来表示
绝对转移和调用指令	在 AT&T 汇编格式中，绝对转移和调用指令（jump/call）的操作数前要加上“*”作为前缀	在 Intel 格式中则不需要
远程转移指令和远程子调用指令的操作码	在 AT&T 汇编格式中为“ljump”和“lcall”	在 Intel 汇编格式中则为“jmp far”和“call far”
内存操作数的寻址方式	section:disp(base, index,scale)	section:[base + index*scale+disp]

应用实例如下。

使用默认汇编方式产生名为 hello.o 的目标文件。

AT&T 格式文件代码如下：

```
#hello.s
.data                                           # 数据段声明
      msg : .string "Hello, world!\\n"          # 要输出的字符串
      len = . - msg                             # 字串长度
.text                                           # 代码段声明
.global _start                                  # 指定入口函数
_start:                                         # 在屏幕上显示一个字符串
      movl $len, %edx                           # 参数三：字符串长度
```

```
        movl $msg, %ecx                      # 参数二：要显示的字符串
        movl $1, %ebx                        # 参数一：文件描述符(stdout)
        movl $4, %eax                        # 系统调用号(sys_write)
        int  $0x80                           # 调用内核功能
                                             # 退出程序
        movl $0,%ebx                         # 参数一：退出代码
        movl $1,%eax                         # 系统调用号(sys_exit)
        int  $0x80                           # 调用内核功能
```

使用命令：

```
# as -o hello.o hello.s
```

汇编程序在成功通过 GAS 或 NASM 的编译并生成目标代码后，就可以使用 ld 命令将其链接成可执行程序了。

```
#ld -s -o hello hello.o
```

运行程序。

```
# ./hello
Hello, world!\n
```

12.2 nasm：汇编器

作用：nasm 命令是一个为可移植与模块化而设计的 80x86 的汇编器，它支持相当多的目标文件格式。

用法：nasm -f <format> <filename>[选项] [-o <output>]

主要选项如下。

- -f <format>：指定输出文件的格式。
- -F stabs：生成调试信息的格式，对于 gcc 来说都是用 stabs。
- -g：激活调试信息。
- -M：产生 Makefile 依赖关系。
- -g：使调试信息有效。
- -E：把错误信息输入到文件。
- -s：把错误信息输出到'stdout'。
- -I：包含文件搜索路径。
- -p：预包含一个文件。
- -d：预定义一个宏。
- -u：取消一个宏定义。
- -e：仅预处理。

- -a：不需要预处理。
- On：指定多遍优化。
- -l：产生列表文件。

应用实例如下。

以 hello.asm 这个小程序为例，介绍如何用 nasm 在 Linux 下进行汇编。

```
hello.asm
hello world for linux
section .text
extern puts
global main
main:
   push dword msge ;
   call puts ;
   add  esp, byte 4 ;
   ret ;

msge:
   db "hello world!",0
```

nasm 编译命令如下：

```
#nasm -f elf hello.asm
gcc -o hello hello.o
```

“nasm -f elf hello.asm”将把 hello.asm 汇编成 elf object 文件。“gcc -o hello hello.o”将把 hello.asm 汇编成二进制可执行文件 hello.com。

nasm -f 命令的参数主要有 aout 和 elf 两种。如果不能确定 Linux 系统应该用 aout 还是 elf，可以在 nasm 目录中输入命令 file nasm，如果输出“nasm：elf 32-bit lsb executable i386(386 and up)version1”，则应该使用 elf；如果输出“nasm：linux/i386 demand-paged executable(qmagic)”，则应该使用 aout。

12.3 gcc：C/C++编译器

作用：gcc 是自由软件基金会（Free Software Foundation）推出的多平台编译器。

用法：gcc [选项]文件...

主要选项如下。

- -pass-exit-codes：在某一阶段退出时返回最高的错误码。
- --help：显示此帮助说明。

- --target-help：显示目标机器特定的命令行选项（使用“-v--help”显示子进程的命令行参数）。
- -dumpspecs：显示所有内建的 spec 字符串。
- -dumpversion：显示编译器的版本号。
- -dumpmachine：显示编译器的目标处理器。
- -print-libgcc-file-name：显示编译器伴随库的名称。
- -print-search-dirs：显示编译器的搜索路径
- -print-file-name=<库>：显示<库>的完整路径。
- -print-prog-name=<程序>：显示编译器组件<程序>的完整路径。
- -print-multi-directory：显示不同版本 libgcc 的根目录。
- -print-multi-lib：显示命令行选项和多个版本库搜索路径间的映射。
- -print-multi-os-directory：显示操作系统库的相对路径。
- -Wa,<选项>：将逗号分隔的<选项>传递给汇编器。
- -Wp,<选项>：将逗号分隔的<选项>传递给预处理器。
- -Wl,<选项>：将逗号分隔的<选项>传递给链接器。
- -Xassembler <参数>：将<参数>传递给汇编器。
- -Xpreprocessor <参数>：将<参数>传递给预处理器。
- -Xlinker <参数>：将<参数>传递给链接器。
- -combine：将多个源文件一次性传递给汇编器。
- -save-temps：不删除中间文件。
- -pipe：使用管道代替临时文件。
- -time：为每个子进程计时。
- -specs=<文件>：用<文件>的内容覆盖内建的 specs 文件。
- -std=<标准>：指定输入源文件遵循的标准。
- --sysroot=<目录>：将<目录>作为头文件和库文件的根目录。
- -B<目录>：将<目录>添加到编译器的搜索路径中。
- -b <机器>：为 gcc 指定目标机器（如果有安装）。
- -V <版本>：运行指定版本的 gcc（如果有安装）。
- -v：显示编译器调用的程序。
- -###：与-v 类似，但选项被引号括住，并且不执行命令。
- -E：仅做预处理，不进行编译、汇编和链接。
- -S：编译到汇编语言，不进行汇编和链接。
- -c：编译、汇编到目标代码，不进行链接。
- -o <文件>：输出到<文件>。
- -x <语言>：指定其后输入文件的语言。

以-g、-f、-m、-O、-W 或--param 开头的选项将由 gcc 自动传递给其调用的不同子进程。若要向这些进程传递其他选项，必须使用“-W<字母>”选项。

gcc 支持的基本语言包括 C（基本语言）、C++（第一个附加语言）、Objective-C、Fortran、Java、Ada。

1．gcc 的组成

gcc 是由许多组件组成的。表 12-2 列出了 gcc 的最重要部分，但它们也并不总是出现的，有些部分是和语言相关的。所以如果没有安装某种特定语言，系统中就不会出现相关的文件。

表 12-2　gcc 的组成

组　件	描　述
c++	gcc 的一个版本，默认语言设置为 C++，而且在连接的时候自动包含标准 C++库
cc1	实际的 C 编译程序
cc1plus	实际的 C++编译程序
collect2	在不使用 GNU 连接程序的系统上，有必要运行 collect2 来产生特定的全局初始化代码（例如 C++的构造函数和析构函数）
configure	gcc 源代码树根目录中的一个脚本，用于设置配置值和创建 gcc 编译程序必需的 make 程序的描述文件
f77	该驱动程序可用于编译 Fortran
f771	实际的 Fortran 编译程序
g++	gcc 的一个版本，默认语言设置为 C++，而且在连接的时候自动包含标准 C++库。这和 C++一样
gcc	该驱动程序等同于执行编译程序和连接程序以产生需要的输出
gcj	该驱动程序用于编译 Java
gnat1	实际的 Ada 编译程序
gnatbind	一种工具，用于执行 Ada 语言绑定
gnatlink	一种工具，用于执行 Ada 语言连接
jc1	实际的 Java 编译程序
libgcc	该库包含的例程被作为编译程序的一部分，是因为它们可被连接到实际的可执行程序中。它们是特殊的例程，连接到可执行程序来执行基本的任务，例如浮点运算。这些库中的例程通常都是平台相关的
libgcj	运行时库包含所有的核心 Java 类
libobjc	对所有 Objective-C 程序都必需的运行时库
libstdc++	运行时库包括定义为标准语言一部分的所有的 C++类和函数

2．运行 gcc 时调用的程序和命令

表 12-3 列出的软件和 gcc 协同工作，目的是实现编译过程。有些是很基本的（例如 as 和 ld），而其他一些则是非常有用但不是严格需要的。这些工具中很多都是各种 UNIX 系统的本地工具。

表 12-3　gcc 使用的软件工具

工　具	描　述
addr2line	给出一个可执行文件的内部地址，addr2line 使用文件中的调试信息将地址翻译成源代码文件名和行号
ar	这是一个程序，可通过从文档中增加、删除和析取文件来维护库文件。通常使用该工具是为了创建和管理连接程序使用的目标库文档

续表

工　具	描　述
as	GNU 汇编器。实际上它是一族汇编器，因为它可以被编译或能够在各种不同的平台上工作
autoconf	产生的 Shell 脚本自动配置源代码包去编译某个特定版本的 UNIX
f2c	是 Fortran 到 C 的翻译程序
gcov	gprof 使用的配置工具，用来确定程序运行的时候哪一部分耗时最大
gdb	GNU 调试器，可用于检查程序运行时的值和行为
gprof	该程序会监督编译程序的执行过程，并报告程序中各个函数的运行时间，可以根据所提供的配置文件来优化程序
ld	GNU 连接程序。该程序将目标文件的集合组合成可执行程序
libtool	一个基本库，支持 make 程序的描述文件使用的简化共享库用法的脚本
make	一个工具程序，它会读 makefile 脚本来确定程序中的哪部分需要编译和链接，然后发布必要的命令。它读出的脚本（叫作 makefile 或 Makefile）定义了文件关系和依赖关系
nm	列出目标文件中定义的符号
objcopy	将目标文件从一种二进制格式复制和翻译到另外一种
objdump	显示一个或多个目标文件中保存的多种不同信息
ranlib	创建和添加到 ar 文档的索引。该索引被 ld 用来定位库中的模块
size	列出目标文件中每个部分的名字和尺寸
strings	浏览所有类型的文件，析取出用于显示的字符串
strip	从目标文件或文档库中去掉符号表，以及其他调试所需的信息

3．分步执行 gcc

在使用 gcc 编译程序时，编译过程可以被细分为 4 个阶段。

1）预处理（Pre-Processing）

gcc 会调用 cpp 程序来分析各种宏指令，如#define、#if、#include 等。

```
# gcc -E hello.c -o hello.cpp
```

-E 参数可以让 gcc 在预处理结束后停止编译过程，此时若查看 hello.cpp 文件中的内容，会发现 stdio.h 的内容确实都插到文件里去了，而其他应当被预处理的宏定义也都做了相应的处理。可以通过使用-o 编译选项，告诉 gcc 我们想如何命名该文件。编译器还必须知道用户期望获得哪种类型的输出。gcc 驱动器可以为整个工具链的其他部分产生一些指令，从而生成最终的可执行程序，或者在生成一些中间文件时就停止。可以使用源程序的文件扩展名或命令行选项来控制编译器的输出。表 12-4 对输出文件的选项进行了总结。

表 12-4　输出文件的选项

标　记	操　作	输出格式
-c	汇编，但不链接	.o 文件
-S	编译，但不汇编	.s 文件
-E	预处理，但不编译	对.c 文件进行预处理后的结果

2）编译（Compiling）

这一阶段根据输入文件产生汇编语言指令。由于通常情况下是立即调用汇编程序 as，所以输出一般不保存在文件中，可以使用-c 选项完成。

```
# gcc -x cpp-output -c hello.cpp -o hello.o
```

-x 选项告诉 gcc 从指定的步骤开始编译。gcc 通过依靠文件扩展名来决定如何处理该文件。表 12-5 列出了 gcc 可以处理的主要文件扩展名及解释类型。

表 12-5　gcc 对文件扩展名的解释类型

扩 展 名	类　　型
.c	C 语言源代码
.C、.cc	C++语言源代码
.i	预处理后的 C 源代码文件
.ii	预处理后的 C++源代码文件
.S、..s	汇编语言源代码
.o	目标文件（Object File）
.m	Objective-C 源代码文件
.a	归档库文件（Archive File）
.h	预处理器文件

3）汇编（Assembling）

这一阶段将汇编语言源程序作为输入，生成.o 目标文件。下面的命令将生成的目标文件链接成可执行文件：

```
#  gcc hello.o -o hello
```

4）链接（Linking）

这是最后一个阶段。在该阶段中，各个.o 模块被链接在一起构成可执行文件。实际工作中整个程序是由多个源文件组成的，相应地形成了多个编译单元，使用 gcc 能够很好地管理这些编译单元。本文中这个部分因为只有一个源程序而省略。图 12-1 所示是 gcc 处理程序的示意图。

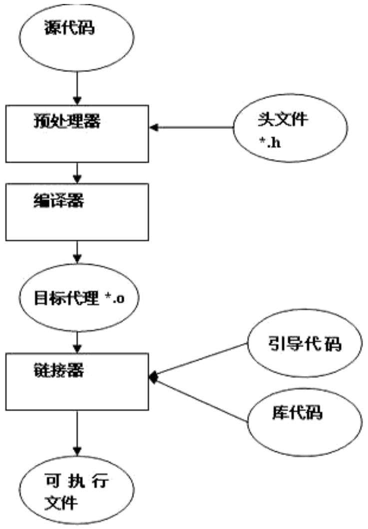


图 12-1　gcc 编译过程示意图

4．gcc 警告提示功能

gcc 包含完整的出错检查和警告提示功能，可以帮助 Linux 程序员写出更加专业的代码。表 12-6 列出了常见的警告选项。

表 12-6 常见的警告选项

选 项	说 明
-Wcomment	如果注释起始序列"/*"出现在注释中，编译器就发出警告
-Wformat	检查对 printf 和 scanf 等函数的调用，确认各个参数类型和格式串中的一致
-Wmain	如果把 main 函数声明或定义成非法的类型，编译器就发出警告
-Wparentheses	在某些情况下如果忽略了括号，编译器就发出警告
-Wswitch	如果某条 switch 语句的参数属于枚举类型，但是没有对 case 语句使用枚举元素，编译器就发出警告，超出枚举范围的 case 语句同样会导致这个警告
-Wunused	如果声明了一个变量却没再使用，编译器就发出警告
-Wuninitialized	在初始化之前就使用自动变量，编译器就发出警告
-Winline	如果某个函数不能内嵌（inline），编译器将发出警告
-Wmissing-declarations	如果没有预先声明就定义了全局函数，编译器就发出警告
-Wlong-long	如果使用了 long long 类型，编译器就发出警告。该警告是默认项，使用"-Wno-long-long"选项能够防止这个警告出现
-Werror	在发生警报时取消编译操作，即将警报看作错误，在使用自动编译工具（如 Make 等）时非常有用。编译时带上-Werror 选项，那么 gcc 会在所有产生警告的地方停止编译，迫使对代码进行修改。只有当相应的警告信息消除时，才能继续编译过程

应用实例如下。

1）一个简单例子

首先从一个最简单的例子开始，能够帮助用户迅速理解 gcc 的工作原理，并将其立即运用到实际的项目开发中去。首先用编辑器（vi 或者 Emacs）输入如下代码。

```
hello.c:
#include <stdio.h>
int main(void)
{
   printf ("Hello Linux!\n");
   return 0;
}
```

然后执行下面的命令编译和运行这段程序。

```
# gcc hello.c -o hello
# ./hello
Hello Linux!
```

第一行命令告诉 gcc 对源程序 hello.c 进行编译和链接，并且使用-o 参数指定创建名称是 hello 的可执行程序；第二行命令执行 hello 整个程序；第三行是程序的执行结果。整个过程看起来一气呵成，但是实际上这段时间内还发生了读者没有看到的事情。首先，gcc 需要调用预

处理程序 cpp，由它负责展开在源文件中定义的宏，并向其中插入“#include”语句所包含的内容；接着，gcc 会调用 as 将处理后的源代码编译成目标代码；最后，gcc 会调用链接程序 ld，把生成的目标代码链接成一个可执行程序。

2）使用 gcc 处理多个文件

假设有一个由 file1.c、file2.c 和 file3.c 3 个源文件组成的程序，为了对它们进行编译，并最终生成可执行程序 progname，可以使用下面这条命令：

```
$ gcc file1.c file2.c file3.c -o progname
```

如果同时处理的文件不止一个，gcc 仍然会按照预处理、编译和链接的过程依次进行。如果深究起来，上面这条命令大致等同于依次执行如下命令：

```
$ gcc -c file1.c
$ gcc -c file2.c
$ gcc -c file3.c
$ gcc file1.o file2.o file3.o -o progname
```

多数 C 程序是由多个源代码文件组成的，所以必须把源代码文件编译成为目标代码才能进行链接。gcc 可以轻松实现这一目标。假设一个 howdy（C 源代码名称为 howdy.c）程序使用了来自 helper.c 的代码和头文件 helper.h。程序清单如下：

```
helper.c
 /*
 * helper.c - Helper code for howdy.c
 */

#include
#include "helper.h"

void msg(void)
{
    printf("This message sent from Jupiter.\n");
}
helper.h
/*
 * helper.h - Header for helper.c
 */
void msg(void);

howdy.c
/*
 * hello.c - Canonical "Hello, World!" program
```

```
 */
#include
#include "helper.h"

int main(void)
{
    printf("Hello, Linux programming world!\n");
    msg();
    return 0;
}
```

编译后就可以执行这个命令。

```
#gcc howdy.c helper.c -o howdy
# ./howdy
Hello, Linux programming world!
This message sent from Jupiter.
```

3）使用 gcc 编译 Fortran 程序

传统的 Fortran 程序可完全用大写字母编写，每行的前 6 个字符位置留空，用于其他特殊用途。留出第一列为字符 C 说明该行是注释，从第二列到第六列留作标号（labels），代码从第七列开始。下面的例子是传统格式的 Fortran 程序。

```
PROGRAM HELLOWORLD
     WRITE(*,10)
    10 FORMAT('hello, world')
     END PROGRAM HELLOWORLD
```

编译器 gfortran 并不要求所有代码都大写——程序中任何关键词都可以用小写字母。下面的命令将该程序编译成可执行文件。

```
$ gfortran helloworld.f -o helloworld
```

说明

gfortran 默认会将以.f、.for、.fpp、.ftn、.F、.FOR、.FPP 和.FTN 结尾的文件作为固定格式处理，而将以.f90、.f95、.f03、.F90、.F95 和.F03 结尾的文件作为自由格式来处理。如果我们将上面的程序文件重命名为 helloworld.f90，那么必须手动指定其为固定格式。

```
$ mv helloworld.f helloworld.f90
$ gfortran helloworld.f90 -ffixed-form -o helloworld
```

Fortran 90 及以后的标准允许并鼓励用自由的格式书写 Fortran 代码。注释以感叹号（!）开始直到行尾。先前的程序采用自由格式重写如下，其中语句、标号都可从任意一列开始。

```
! helloworldff.f90
!
Program Helloworld
write(*,10)
10 format('hello, world')
end
```

Program Helloworld 扩展名为.f90，故 gfortran 将其作为自由格式处理：

```
$ gfortran helloworldff.f90 -o helloworldff
```

同样，如果将程序重命名为传统扩展名，那么要通过在命令行中加入选项“-ffree-form”进行编译。

```
$ mv helloworldff.f90 helloworldff.for
$ gfortran -ffree-form helloworldff.for -o helloworldff
```

由于两种格式具有很大的区别，所以只能选择其中的一种格式进行书写。注意，遵守扩展名约定是很重要的。

12.4 make：维护和编译软件或软件包

作用：make 命令是维护和编译软件或软件包的工具。

用法：make [选项] [目标]

主要选项如下。

- -B, --always-make：无条件执行所有目标文件。
- -b-m：忽略和其他版本 make 的兼容性。
- -f file：指定 file 文件为描述文件。如果 file 参数为“-”，那么描述文件指向标准输入。如果没有“-f”参数，则系统将默认当前目录下名为 makefile 或者 Makefile 的文件为描述文件。
- -i：忽略命令执行返回的出错信息。
- -s：沉默模式，在执行之前不输出相应的命令行信息。
- -r, --no-builtin-rules：禁止 make 使用任何隐含规则。
- -R, --no-builtin-variables：禁止 make 使用任何作用于变量上的隐含规则。
- -o <file>--old-file=<file>--assume-old=<file>：不重新生成指定的<file>，即使这个目标的依赖文件比它新。
- -n：非执行模式，输出所有执行命令，但并不执行。
- -t：更新目标文件。

- -q：不运行命令，也不输出，仅仅检查所指定的目标是否需要更新。如果是 0，则说明要更新；如果是 2，则说明有错误发生。
- -p：输出所有宏定义和目标文件描述。
- -d：Debug（调试）模式，输出有关文件和检测时间的详细信息。
- -t, --touch：相当于 Linux 的 touch 命令，只是把目标的修改日期变成最新的，也就是阻止生成目标的命令运行。
- -v,--version：输出 make 程序的版本、版权等关于 make 的信息。
- -C DIR：在读取 makefile 之前改变到指定的目录。
- -h：显示所有的 make 选项。
- -w：在处理 makefile 之前和之后，显示工作目录。
- -W file：假定文件 file 已经被修改。使用--n 来显示 make 要执行的命令。
- --jN：同时运行多个 make 命令，即并行处理程序。其中 N 是非零整数。
- --warn-undefined-variables：只要 make 发现有未定义的变量，就输出警告信息。
- -l <load>--load-average[=<load]--max-load[=<load>]：指定 make 运行命令的负载。

1．调试 make

```
--debug[=<options>]
```

输出 make 的调试信息。它有几种不同的级别可供选择，如果没有参数，则输出最简单的调试信息。<options>的取值如下。

- a：也就是 all，输出所有的调试信息。
- b：也就是 basic，只输出简单的调试信息，即输出不需要重编译的目标。
- v：也就是 verbose，在 b 选项的级别之上。输出的信息包括哪个 makefile 文件被解析、不需要被重编译的依赖文件（或依赖目标）等。
- i：也就是 implicit，输出所有的隐含规则。
- j：也就是 jobs，输出执行规则中命令的详细信息，如命令的 PID、返回码等。
- m：也就是 makefile 文件，输出 make，读取 makefile 文件，更新 makefile 文件，执行 makefile 文件的信息。

2．make 的退出码

make 命令执行后有以下 3 个退出码。

- 0：表示成功执行。
- 1：如果 make 运行时出现错误，其返回值为 1。
- 2：如果使用了 make 的-q 选项，并且 make 使得一些目标不需要更新，则返回 2。

应用实例如下。

从一个简单 makefile 文件理解 make。

上面介绍的是理论知识，下面介绍一个简单的 makefile 文件。makefile 文件程序如下：

```
#a Sample makefile
howdy: howdy.o helper.o helper.h
gcc howdy.o helper.o -o howdy
```

```
helper.o: helper.c helper.h
 gcc -c helper.c
howdy.o: howdy.c
 gcc -c howdy.c
hello: hello.c
 gcc hello.c -o hello
all: howdy hello
clean:
    rm howdy hello *.o
```

makefile 中涉及的文件如下。

helper.c：

```
 /* helper.c - Helper code for howdy.c */
#include
#include "helper.h"
void msg(void)
{
    printf("This message sent from Jupiter.\n");
}
```

helper.h：

```
/* helper.h - Header for helper.c */
void msg(void);
```

howdy.c：

```
/* hello.c - Canonical "Hello, World!" program */
#include
#include "helper.h"
int main(void)
{
    printf("Hello, Linux programming world!\n");
    msg();
    return 0;
}
```

hello.c：

```
/* hello.c - Canonical "Hello, World!" program */
#include <stdio.h>
int main(void)
{
```

```
    printf("Hello, Linux programming world!\n");

    return 0;
}
```

把以上 5 个文件复制到一个目录下，使用 make 命令就可以看到执行过程，如图 12-2 所示。

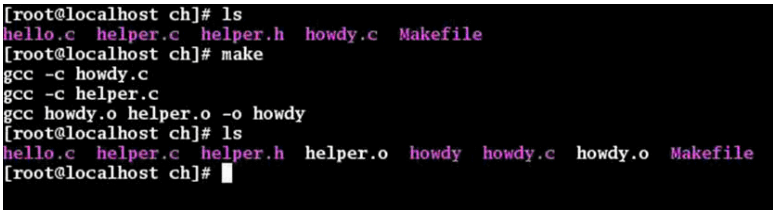


图 12-2　make 执行过程

下面我们使用文字详细解释执行过程。

makefile 文件包括 6 个规则。第一个目标体 howdy 是默认的目标体，即 make 命令要创建的文件。howdy 有 3 个依赖体：howdy.o、0helper.o、helper.h。make 命令要编译生成 howdy 文件，这 3 个文件必须存在。第二行调用 gcc 编译器的命令供 make 执行来创建 howdy。这条命令从两个目标（howdy.o 和 helper.o）文件创建名称为 howdy 的可执行文件。helper.h 头文件作为依赖体的作用是避免编译器调用上面的函数产生错误信息。下面的两个规则告诉 make 如何生成目标文件 howdy.o 和 helper.o。这里使用 gcc 命令的-c 选项。如果只想生成两个目标文件，可以使用命令“make howdy.o helper.o”。make 允许一次使用多个目标作为参数。如图 12-3 所示是从一个 makefile 建立可以执行文件的示意图。

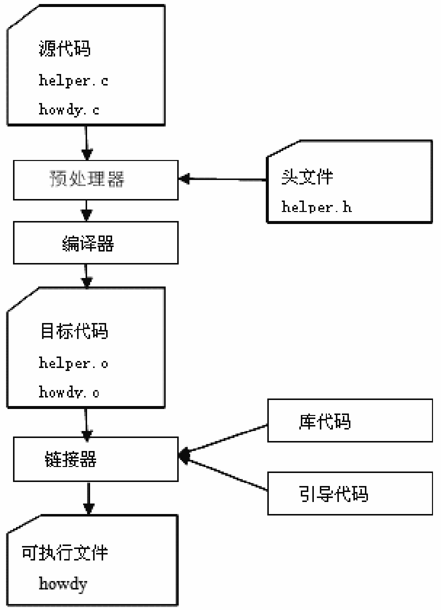


图 12-3　从一个 makefile 建立可以执行文件

图 12-3 中显示：首先把 howdy.c 和 helper.c 两个源代码文件经过预处理后编译成为目标文件。然后链接器把文件 howdy.o、helper.o 和标准库，以及 C 语言启动代码链接到以前生成的二进制文件 hello。第四个规则定义如何编译 gcc 生成程序。第五个规则是创建 hello 和 howdy 的笼统规则，它表明二进制文件也可以作为依赖体。第六个规则是伪目标（后面将具体讲解）。从如图 12-2 所示的两个 ls 命令也可以看到 make 执行前后的文件增加情况。

现在总结介绍一下 make 的工作原理。默认 make 从第一个 target 开始（第一个非“.”开始的 target），这称为默认目标。在上面的 makefile 中，默认目标是更新执行程序“howdy”，将这个目标置于最前面。当执行 make 的时候，make 程序从当前目录读入 makefile 并开始处理第一个规则；在这一例子中，这个规则是重新链接“howdy”；在 make 处理这个规则之前，必须处理“howdy”所依赖的那些文件的规则，例子中是目标文件。这些文件按照它们自己的规则处理：通过编译源文件来更新每个“.o”文件；当依赖关系中的源文件或头文件比目标文件新，或目标文件不存在时，必须重新编译。

其他的规则被处理是因为它们的 target 是目标的依赖，和目标没有依赖关系的规则不会被处理，除非指定 make 处理（如 make clean）。在重新编译目标文件之前，make 会试图更新它的依赖：源文件和头文件。这一例子中的 makefile 对源文件和头文件未指定任何操作：“.c”和“.h”文件不是任何规则的目标。确认所有的目标文件都是最新的之后，make 决定是否重新链接“howdy”：如果“howdy”不存在，或者任何一个目标文件都比它新，则链接工作将进行。这样，如果我们改变任何一个源代码文件运行 make，make 就会编译这个文件来更新目标文件，然后链接“howdy”。make 的魅力就在于能够自动进行条件检测，并采取适当的行动。它永远也不会去编译那些没有改动过的源文件，因此大大节省了在开发大型软件项目时浪费在编译上的时间。

12.5　gdb：GUN 调试器

作用：gdb 命令是 GUN 调试器。该命令行调试器可用于全面控制和检查运行中的程序。

用法：gdb 内部命令

gdb 支持很多的命令，可以实现不同的功能。表 12-7 列出了使用 gdb 调试时会用到的一些命令。

表 12-7　基本 gdb 命令

命　　令	描　　述
file	装入想要调试的可执行文件
kill	终止正在调试的程序
list	列出产生执行文件的源代码的一部分
next	执行一行源代码但不进入函数内部
step	执行一行源代码而且进入函数内部

续表

命　　令	描　　述
run	执行当前被调试的程序
quit	终止 gdb
watch	使用户能监视一个变量的值而不管它何时被改变
break	在代码里设置断点，这将使程序执行到这里时被挂起
make	使用户能不退出 gdb 就可以重新产生可执行文件
shell	使用户能不离开 gdb 就可以执行 UNIX Shell 命令

gdb 支持很多与 UNIX Shell 程序一样的命令编辑特征。用户能像在 bash 或 tcsh 里那样按 Tab 键让 gdb 帮助补齐一个唯一的命令，如果不唯一的话，gdb 会列出所有匹配的命令。也可以用光标键上下翻动历史命令。

说明

工具程序 gdb 是 GNU 调试器，该命令行调试器可用于全面控制和检查运行中的程序。所有程序都会对 gdb 发出的命令有所响应，但只有按照合适选项编译并连接的程序才能包含足够信息，而这些信息才能够连接到源文件上，用于跟踪程序流程。启动交互调试过程最简单的方法就是在命令行中直接指出将由 gdb 启动该程序，当然这和先启动调试器然后再载入程序的结果一样。也可指出要将调试器自己附带到运行程序上，这样只有在程序运行一次以后，才可能对程序内部的异常处理过程进行检查。使用 gdb 的另一个目的是为了对已崩溃的程序进行事后检查，以确定崩溃的原因。

gdb 主要具有以下 4 个方面的功能。

– 启动程序，可以按照自定义的要求随心所欲地运行程序。
– 可让被调试的程序在所指定的调置的断点处停止（断点可以是条件表达式）。
– 当程序被停止时，可以检查此时程序中所发生的情况。
– 动态地改变程序的执行环境。

从上面看来，gdb 和一般的调试工具没有什么两样，基本上也是完成这些功能，不过在细节上，你会发现 gdb 这个调试工具的强大。大家可能习惯了图形化的调试工具，但有时候，命令行的调试工具却有着图形化的调试工具所不能完成的功能。

应用实例如下。

下面还是通过一个具体的实例说明如何利用调试符号来分析错误，所用程序如下：

```
 /* debugme.c - Poorly written program to debug*/
#include
#include
#define BIGNUM 5000
void index_to_the_moon(int ary[]);
int main(void)
```

```
{
    int intary[100];
    index_to_the_moon(intary);
    exit(EXIT_SUCCESS);}
void index_to_the_moon(int ary[])
{
int i;
for(i = 0; i < BIGNUM; ++i)
    ary[i] = i;}
```

如果想对程序进行调试，必须在用 gcc 编译源代码时加上-g 选项，以便产生 gdb 所需要的调试符号信息。例如，debugme.c 是一个存在错误的程序，可以使用如下命令对其进行编译，同时产生调试符号。

```
# gcc -g debugme.c -o debugme
```

还可以在编译时使用“-ggdb”选项来生成更多的调试信息。由于这些调试信息中相当大的一部分是 gdb 所特有的，所以生成的代码将无法在其他调试器中正常调试。对于大多数情况来说，普通的-g 选项就足够了。需要注意的是，gcc 虽然允许同时使用-g（调试）和-o（优化）选项，但优化会影响最终生成的代码，导致程序源代码和二进制代码之间的关系变得复杂起来。如果不想为调试制造障碍，建议不要将-g 和-o 选项一同使用，并且只在程序彻底调试完后才开始进行代码优化，这样调试过程将变得相对轻松和愉快。

1）基本应用

现在可以启动 gdb 来调试已经生成的可执行程序 debugme，命令操作如下：

```
# gdb debugme   <---------- 启动 GDB
GNU gdb Fedora (6.8-37.el5)
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i386-redhat-linux-gnu"...
(gdb) f debugme
No symbol "debugme" in current context.
(gdb) file debugme
Load new symbol table from "/tmp/debugme"? (y or n) y
Reading symbols from /tmp/debugme...done.
(gdb) l <-------------------- l 命令相当于 list，从第一行开始列出原码
6
7      #define BIGNUM 5000
```

```
8
9       void index_to_the_moon(int ary[]);
10
11      int main(void)
12      {
13              int intary[100];
14
15              index_to_the_moon(intary);
(gdb) b 10 <-------------------- 设置断点，在源程序第 10 行处
Breakpoint 1 at 0x8048384: file debugme.c, line 10.
(gdb) info break <-------------------- 查看断点信息
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x08048384 in main at debugme.c:10
(gdb) run <--------------------- 运行程序，run 命令简写
Starting program: /tmp/debugme
Breakpoint 1, main () at debugme.c:12
12      { <---------- 在断点处停止
(gdb) n <----------------- 单条语句执行，next 命令简写
main () at debugme.c:15
15              index_to_the_moon(intary);
(gdb) c <--------------- 继续运行程序，continue 命令简写
Continuing.
Program received signal SIGSEGV, Segmentation fault.
0x080483cf in index_to_the_moon (ary=0xbfb3e504) at debugme.c:24
24                      ary[i] = i;
(gdb) p i  <---打印变量 i 的值，print 命令简写
$1 = 703
(gdb) bt <--------------------- 查看函数堆栈
#0  0x080483cf in index_to_the_moon (ary=0xbfb3e504) at debugme.c:24
#1  0x080483a6 in main () at debugme.c:15
(gdb) c <--------------------- 继续运行
Continuing.
Program terminated with signal SIGSEGV, Segmentation fault.
The program no longer exists.
(gdb) q <--------------------- 退出 gdb
```

最后一行输出表明程序在调用动态链接库/lib/ld-linux.so.2 中的_dl_fini()函数时出现了错误，地址是 0x4000c6ac。这些对调试是非常重要的线索。另外还有一种信息对调试也很重要，就是错误发生时的函数调用层级关系，可以通过执行“backtrace”命令来获得。在使用 gdb 调

试命令时，用户可以不必输入完整的命令名称，使用任何唯一的缩写都可以。例如“backtrace”命令就可以缩写成“back”甚至“bt”。gdb 还支持很多常用的 Shell 命令编辑特征，比如可以像在 bash 或 tcsh 中那样按 Tab 键补齐命令。如果相关命令不唯一的话，则列出所有可能的匹配项。此外，键盘上的方向键可用来翻动历史命令。

gdb 是一个源代码级的调试器，使用“list”命令可以查看当前调试对象的源代码。该命令的通用格式为“list [m,n]”，表示显示从 *m* 行开始到 *n* 行结束的代码段，而不带任何参数的“list”命令将显示最近 10 行源代码。

说明

gdb 将被启动并在屏幕上输出版权信息，但如果使用了-q 或--quiet 选项，则不会显示它们。启动 gdb 时，另外一个有用的命令行选项是“-d dirname”，其中 dirname 是一个目录名。该目录名告诉 gdb 应该到哪里去寻找源代码。一旦出现 gdb 的命令提示符（gdb），就表明 gdb 已经准备好接收来自用户的各种调试命令了。如果想在调试环境下运行这个程序，可以使用 gdb 提供的“run”命令，而程序在正常运行时所需的各种参数可以作为“run”命令的参数传入，或者使用单独的“set args”命令进行设置。如果在执行“run”命令时没有给出任何参数，gdb 将使用上一次“run”或“set args”命令指定的参数。如果想取消上次设置的参数，可以执行不带任何参数的“set args”命令。下面尝试在调试器中运行这个程序：

```
(gdb) run
…
Program received signal SIGSEGV, Segmentation fault.
0x4000c6ac in _dl_fini () from /lib/ld-linux.so.2
```

2）设置断点（Breakpoint）

断点是调试器控制程序执行的基本手段。设置断点就是用机器断点指令替换所指定程序单元中的指令，使得程序运行到断点指令处时，产生“断点异常”，用户程序不再继续执行下去，目标机向宿主机返回断点停止信号，由宿主机调试器接管对用户程序的控制。在调试有问题的代码时，在某一点停止运行往往很有用。这样程序运行到此处时会暂时挂起，等待用户的进一步输入。gdb 允许在几种不同的代码结构上设置断点，包括行号和函数名等，并且还允许设置条件断点，让程序只有在满足一定的条件时才停止执行。要根据行号设置断点，可以使用“break linenum”命令。要根据函数名设置断点，则应该使用“break funcname”命令。

在以上两种情况中，gdb 将在执行指定的行号或进入指定的函数之前停止执行程序。此时可以使用“print”显示变量的值，或者使用“list”查看将要执行的代码。对于由多个源文件组成的项目，如果想在执行到非当前源文件的某行或某个函数时停止执行，可以使用如下形式的命令：

```
# break filename:linenum
# break filename:funcname
```

条件断点允许当一定条件满足时，暂时停止程序的执行。它对于程序调试非常有用。设置条件断点的正确语法如下：

```
break linenum if expr
break funcname if expr
```

其中 expr 是一个逻辑表达式。当该表达式的值为真时，程序将在该断点处暂时挂起。例如，下面的命令将在 debugme 程序的第 38 行设置一个条件断点。当程序运行到该行时，如果 count 的值等于 3，就暂时停止执行。

```
(gdb) break 38 if count==3
```

设置断点是调试程序时最常用到的一种手段。它可以中断程序的运行，给程序员一个单步跟踪的机会。使用命令“break main”在 main 函数上设置断点，可以在程序启动时就开始进行跟踪。接下来使用“continue”命令继续执行程序，直到遇到下一个断点。如果在调试时设置了很多断点，可以随时使用“info breakpoints”命令来查看设置的断点。此外，开发人员还可以使用“delete”命令删除断点，或者使用“disable”命令来使设置的断点暂时无效。被设置为无效的断点在需要时可以用“enable”命令使其重新生效。

在 gdb 中设置与恢复断点的常见命令如下。

- bnfo b：查看所设断点。
- break：通过行号或函数名<条件表达式>设置断点。
- tbreak：通过行号或函数名<条件表达式>设置临时断点，到达后被自动删除。
- delete [断点号]：删除指定断点，其断点号为“info b”中的第一栏。若省略断点号则删除所有断点。
- disable [断点号]：停止指定断点，使用“info b”仍能查看此断点。同 delete 一样，省略断点号则停止所有断点。
- enable [断点号]：激活指定断点，即激活被 disable 停止的断点。
- condition [断点号] <条件表达式>：修改对应断点的条件。
- ignore [断点号]<num>：在程序执行过程中，忽略对应断点 num 次。
- step：单步恢复程序运行，且进入函数调用。
- next：单步恢复程序运行，但不进入函数调用。
- finish：运行程序，直到当前函数完成返回。
- c：继续执行函数，直到函数结束或遇到新的断点。

3）观察变量

gdb 的特性之一是几乎能够显示被调试程序中的任何表达式、变量或数组的类型和值，并且能够用编写程序所用的语言打印出任何合法表达式的值。查看数据最简单的办法是使用“print”命令，只需在“print”命令后面加上变量表达式，就可以打印出此变量表达式的当前值，示例如下：

```
(gdb) print str
$1 = 0x40015360 "Happy new year!\n"
```

从输出信息中可以看出，输入字符串被正确地存储在了字符指针 str 所指向的内存缓冲区中。除给出了变量表达式的值外，“print”命令的输出信息中还包含变量标号（$1）和对应的内存地址（0x40015360）。变量标号保存着被检查数值的历史记录，如果此后还想访问这些值，就可以直接使用别名而不用重新输入变量表达式。如果想知道变量的类型，可以使用“whatis”命令。

在 gdb 中不仅可以调试所运行的程序，还可以对程序相关的工作环境进行相应的设定，甚至可以使用 Shell 中的命令进行相关的操作，其功能极其强大。gdb 中常见的与工作环境相关的命令如下。

- set args：指定运行时参数，如 set args 2。
- show args：查看设置好的运行参数。
- path dir：设定程序的运行路径。
- show paths：查看程序的运行路径。
- set enVironment var [=value]：设置环境变量。
- show enVironment [var]：查看环境变量。
- cd dir：进入到 dir 目录，相当于 Shell 中的 cd 命令。
- pwd：显示当前工作目录。
- shell command：运行 Shell 的 command 命令。

4）单步执行

为了单步跟踪代码，可以使用单步跟踪命令“step”，它每次执行源代码中的一行。还可以直接按 Enter 键来重复执行前一条命令。除可以用“step”命令来单步运行程序之外，gdb 还提供了另外一条单步调试命令“next”。二者功能非常相似，差别在于如果将要被执行的代码行中包含函数调用，使用 step 命令将跟踪进入函数体内，而使用 next 命令则不进入函数体内。finish 命令一直运行到函数返回，until 命令运行到某一行。

5）gdb 中运行 Linux 的 Shell 程序

在 gdb 环境中，可以直接执行 Linux 的 Shell 命令；如果 Shell 没有定义，那就使用 Linux 的标准 Shell：/bin/sh。另外还有一个 gdb 命令 make，可以在 gdb 中执行 make 命令来重新编译（build）自己的程序，这个命令等价于“shell make”。

6）gdb 中查看源代码的相关命令

在 gdb 中可以查看源代码以方便其他操作，它的常见相关命令如下。

- list <行号>|<函数名>：查看指定位置代码。
- file [文件名]：加载指定文件。
- forward-search：正则表达式，源代码前向搜索。
- reverse-search：正则表达式，源代码后向搜索。
- dir dir：停止路径名。
- show directories：显示定义了的源文件搜索路径。
- info line：显示加载到 gdb 内存中的代码。

12.6 gdbserver：远端 GNU 服务器

作用：gdbserver 命令用来控制远程 gdb 调试。

用法：gdbserver tty COMM --attach PID

主要选项如下。

- --attach PID：指定进程号。
- Tty：终端号。
- COMM：串口号。

应用实例如下。

进行远程调试。

要进行 gdb 调试，首先要在目标系统上启动 gdbserver 服务。在 gdbserver 所在目录下输入如下命令：

```
(minicom 下)
#cd /tmp
#./gdbserver 192.168.2.100:2345 hello
```

192.168.2.100 为宿主机 IP，在目标系统的 2345 端口开启了一个调试进程，hello 为要调试的程序。

出现如下提示：

```
Process /tmp/hello created: pid="80"
Listening on port 2345
(另一个终端下)
#cd /
#export PATH=$PATH:/usr/local/arm-gdb/bin
#arm-linux-gdb hello
(gdb) target remote 192.168.2.223:2345
（192.168.2.223 为开发版 IP）
```

出现如下提示：

```
Remote debugging using 192.168.2.223:2345
[New thread 80]
[Switching to thread 80]
0x40002a90 in ??()
```

同时在 minicom 下提示：

```
Remote debugging from host 192.168.2.100
(gdb)
```

连接成功，这时就可以输入各种 gdb 命令进行程序调试了。

12.7 autoconf：产生配置脚本

作用：autoconf 是一个用于生成可以自动配置的软件源码包，用以适应多种 UNIX 类系统的 Shell 脚本工具。autoconf 需要用到 m4，便于生成脚本。

用法：autoconf [选项] ... [TEMPLATE-FILE]

主要选项如下。

- -v, --verbose：详细报告处理过程。
- -d, --debug：输出调试信息，不删除临时文件。
- -f, --force：考虑所有过时的文件。
- -o, --output=FILE：保存输出到文件（默认是 stdout）。
- -W, --warnings=CATEGORY：报告报警信息。

报告报警信息包括如下内容。

- ‘cross’：交叉编译。
- ‘obsolete’：过时的架构。
- ‘syntax’：不正确的语法。
- ‘all’：所有报警信息。
- ‘no-CATEGORY’：关闭报警信息。
- ‘none’：关闭所有报警。
- ‘error’：出现错误。

12.8 autoheader：为 configure 产生模板头文件

作用：autoheader 命令生成 config.in，前提是 aclocal.m4 和 configure.in 必须生成，它负责生成 config.h.in 文件。该工具通常会从“acconfig.h”文件中复制用户附加的符号定义。

用法：autoheader [选项] ... [TEMPLATE-FILE]

主要选项如下。

- -v, --verbose：详细报告处理过程。
- -d, --debug：输出调试信息，不删除临时文件。
- -f, --force：考虑所有过时的文件。
- -o, --output=FILE：保存输出到文件（默认是 stdout）。
- -W, --warnings=CATEGORY：报告报警信息。

报告报警信息包括如下内容。

- ‘cross’：交叉编译。
- ‘obsolete’：过时的架构。
- ‘syntax’：不正确的语法。

– ‘all’：所有报警信息。
– ‘no-CATEGORY’：关闭报警信息。
– ‘none’：关闭所有报警。
– ‘error’：出现错误。

12.9 autoreconf：更新已经生成的配置文件

作用：如果有多个 autoconf 产生的配置文件，autoreconf 可以保存一些工作，它通过重复运行 autoconf（以及在合适的地方运行 autoheader）以重新产生 autoconf 配置脚本和配置头模板，这些文件保存在以当前目录为根的目录树中。

用法：autoreconf [选项]...[CONFIGURE-AC or DIRECTORY]

主要选项如下。

– -v, --verbose：详细报告处理过程。
– -f, --force：考虑所有过时的文件。
– -o, --output=FILE：保存输出到文件（默认是 stdout）。
– -W, --warnings=CATEGORY：报告报警信息。
– -d, --debug：输出调试信息，不删除临时文件。
– -i：复制辅助文件。
– -s：创建符号链接，而不是复制。
– -m, --make：当可用时，重新运行命令./configure 和 make。

报告报警信息包括如下内容。

– ‘cross’：交叉编译。
– ‘obsolete’：过时的架构。
– ‘syntax’：不正确的语法。
– ‘all’：所有报警信息。
– ‘no-CATEGORY’：关闭报警信息。
– ‘none’：关闭所有报警。
– ‘error’：出现错误。

12.10 autoscan：产生初步的 configure.in 文件

作用：autoscan 命令可以用来为软件包创建 configure.in 文件。autoscan 在以命令行参数中指定的目录为根（如果未给定参数，则以当前目录为根）的目录树中检查源文件。它为通常的轻便问题搜索源文件，并且为那个包创建一个 configure.scan 文件，这个文件就是 configure.in 的前身。

用法：autoscan [选项] ... [SRCDIR]

主要选项如下。

- -d, --debug：输出调试信息，不删除临时文件。
- [SRCDIR]：要扫描的目录。

12.11 autoupdate：更新 configure.in 文件

作用：autoupdate 命令将一个调用 autoconf 宏的旧名称的 configure.in 文件中的宏更新为新的名称。

用法：autoupdate [选项]...[TEMPLATE-FILE...]

主要选项如下。

- -v, --verbose：详细报告处理过程。
- -d, --debug：输出调试信息。
- -f, --force：考虑所有过时的文件。

12.12 automake：自动生成 Makefile.in 的工具

作用：automake 命令是一个从 Makefile.am 文件自动生成 Makefile.in 的工具。

用法：automake [选项]...[Makefile]

主要选项如下。

- -v, --verbose：详细报告处理过程。
- --no-force：当它输出数据的时候更新 Makefile.in 文件。
- -W, --warnings=CATEGORY：报告报警信息。

报告报警信息包括如下内容。

- ‘cross’：交叉编译。
- ‘obsolete’：过时的架构。
- ‘syntax’：不正确的语法。
- ‘all’：所有报警信息。
- ‘no-CATEGORY’：关闭报警信息。
- ‘none’：关闭所有报警。
- ‘error’：出现错误。

12.13 aclocal：生成 aclocal.m4 文件

作用：aclocal 命令通过扫描 configure.ac 或者 configure.in 文件生成 aclocal.m4 文件。

用法：aclocal[选项]

主要选项如下。

- --acdir=DIR：配置文件和目录。
- -I DIR：为.m4 文件添加目录到查找列表。
- --force：总是更新输出文件。
- --output=FILE：输出到文件（默认是 aclocal.m4）。
- --print-ac-dir：打印使用.m4 文件和目录。

12.14 configure：生成 Makefile 文件

作用：configure 脚本运行时可以侦测系统的环境，确定软件安装目录，然后生成 Makefile 文件。

用法：configure[选项]

主要选项如下。

- --bindir=DIR：指定二进制文件的安装位置，这里的二进制文件定义为可以被用户直接执行的程序。
- --program-prefix=PREFIX：指定将被加到所安装程序的名字上的前缀。
- --disable-FEATURE：一些软件可以选择这个选项来为大型选项的编译提供配置，如果默认提供这些特性，可以使用--disable-FEATURE 来禁用它，这里 FEATURE 是特性的名字。
- -enable-FEATURE[=ARG]：相反，一些软件包可能提供了一些默认被禁止的特性，可以使用--enable-FEATURE 来启用它。这里 FEATURE 是特性的名字。一个特性可能会接受一个可选的参数。
- --with-PACKAGE[=ARG]：在自由软件社区里，有使用已有软件包和库的优秀传统。当用 configure 来配置一个源码树时，可以提供其他已经安装的软件包的信息。
- --without-PACKAGE：有时候用户可能不想让自己的软件包与系统已有的软件包交互。

应用实例如下。

下面我们联合使用以上命令。从 make 的第一个例子 hello.c 程序开始。下面的过程简单地说就是编辑 3 个文件：hello.c、configure.in、makefile.am，然后依次执行 aclocal、autoconf、automake--add-missing、/configure、make、/hello，就可以看到一个符合惯例的 makefile 产生，最后将 hello.c 编译通过。下面介绍详细的过程。

（1）复制 hello.c 文件到一个临时目录。

在工作目录下建立一个 hello 目录，用它来存放 hello 程序及相关文件，如在/tmp/下输入如下代码：

```
/* hello.c - Canonical "Hello, World!" program */
#include <stdio.h>
int main(void)
{
    printf("Hello, Linux programming world!\n");
    return 0;
}
```

现在在 tmp 目录下就应该有一个用户自己写的 hello.c 程序了。

（2）生成 configure。

使用 autoscan 命令来帮助我们根据目录下的源代码生成一个 configure.in 的模板文件。

```
$ autoscan
```

执行后在/tmp 目录下会生成一个文件 configure.scan，我们可以将它作为 configure.in 的蓝本。现在将 configure.scan 改名为 configure.in，并且编辑它，按下面的内容修改，可以去除解释的语句。

```
# -*- Autoconf -*-
AC_INIT(hello.c)
AM_INIT_AUTOMAKE(hello, 1.0)
# Checks for programs.
AC_PROG_CC
# Checks for libraries.
# Checks for header files.
# Checks for typedefs, structures, and compiler characteristics.
# Checks for library functions.
AC_OUTPUT(Makefile)
```

（3）执行命令 aclocal 和 autoconf，会产生 aclocal.m4 及 configure 两个文件。

```
$ aclocal ； autoconf
```

（4）使用编辑器手工建立一个 makefile.am 文件，内容如下：

```
AUTOMAKE_OPTIONS=foreign
bin_PROGRAMS=hello
hello_SOURCES=hello.c
```

automake 会根据 makefile.am 文件自动生成 makefile.in。makefile.am 中定义的宏和目标会指导 automake 生成指定的代码。例如，宏 bin_PROGRAMS 将导致编译和链接的目标被生成。

makefile.am 是一种比 makefile 更高层次的规则，只需指定要生成什么目标、它由什么源文件生成、要安装到什么目录等。

（5）运行 automake，执行如下命令：

```
$ automake --add-missing
```

automake 会根据 Makefile.am 文件产生一些文件，包含最重要的 Makefile.in 文件。

（6）执行 configure 生成 Makefile。

```
$ ./configure
```

（7）使用 Makefile 编译代码。

```
$ make
    运行 hello
 $ ./hello
Hello, Linux programming world!
```

操作过程截图如图 12-4 所示。

要注意的是，利用 autoconf 及 automake 所产生的软件包可以在没有安装 autoconf 及 automake 的环境中使用，这是因为 configure 是一个 Shell 脚本，它已被设计为可以在一般的 UNIX Shell 下执行。可以通过图 12-5 查看产生 Makefile 的流程。

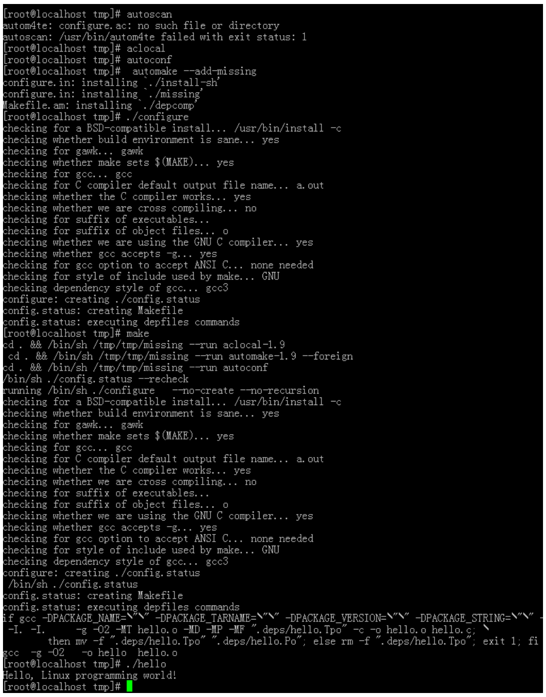


图 12-4　makefile 制作全程截图

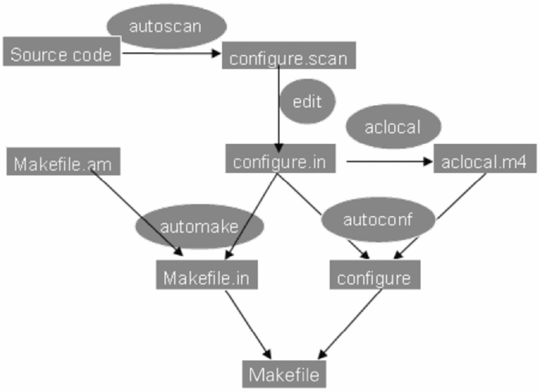


图 12-5　生成 Makefile 的流程图

13 Linux 行编辑器

13.1 ed：文本编辑器

作用：在早期的 UNIX 平台上，ed 可以说是唯一的编辑工具。它是一个很古老的行编辑器，vi 这些编辑器都是从 ed 演化而来的。

用法：ed[-Gs] [-p<字符串>] [--help] [--version] [文件]

主要选项如下。

- -G 或--traditional：支持兼容。
- -p<字符串>：指定 ed 在 command mode 的提示字符。
- -s,-,--quiet 或--silent：不执行开启文件时的检查功能。
- -l：强制退出状态为 0（即使操作中出现错误）。
- -r：在受限模式下运行。

说明

ed 是 Linux 中功能最简单的文本编辑程序，行编辑器使用起来很不方便，一次仅能编辑一行而非全屏幕方式的操作。ed 命令支持多个内置命令，常见内置命令包括：A，切换到输入模式，在文件的最后一行之后输入新的内容；C，切换到输入模式，用输入的内容替换掉最后一行的内容；i，切换到输入模式，在当前行之前加入一个新的空行来输入内容；d，用于删除最后一行文本内容；n，用于显示最后一行的行号和内容；w:<文件名>，以给定的文件名保存当前正在编辑的文件；q，退出 ed 编辑器。

应用实例如下。

编辑 1.txt 文件，支持兼容。

```
# ed -G 1.txt
13909
```

13.2 sed：利用 script 命令处理文本文件

作用：sed 是流编辑器（Streameditor）的缩写。sed 按行对输入文件进行浏览，对输入文件的每行执行指令列表（称为 sed 脚本）。脚本通常都是一个单独的文件，如果只有一行命令，则可包含在 sed 命令行里。

用法：sed [OPTION]...{script-only-if-no-other-script} [input-file]...

主要选项如下。

- -e Script：使用 Script 变量作为编辑脚本。如果只使用一个-e 标志并且不使用-f 标志，则可以省略-e 标志。
- -f ScriptFile：使用 ScriptFile 变量作为编辑脚本源文件。ScriptFile 变量是适用于 File 参数的一组准备好的编辑命令。
- -n：禁止所有正常地写到标准输出的信息。
- -l N,--line-length=N：指定“l”命令的换行期望长度。
- --posix：关闭所有 GNU 扩展。
- -r,--regexp-extended：在脚本中使用扩展正则表达式。
- -s,--separate：将输入文件视为各个独立的文件而不是一个长的连续输入。
- -u,--unbuffered：从输入文件读取最少的数据，更频繁地刷新输出。
- -i[扩展名],--in-place[=扩展名]：直接修改文件（如果指定扩展名就备份文件）。

说明

可以指定多个-e 和-f 标志。按指定的顺序将所有的子命令添加到脚本中，而不管其来源。

sed 命令包含下列 sed 脚本子命令。

- a\：在当前行上添加一个文本行或多个文本行。
- c\：用新文本改动（取代）当前行里的文本。
- d：删除行。
- i\：在当前行之前插入文本。
- h：把模式空间内容复制到一个固定缓存。
- H：把模式空间内容添加到一个固定缓存。
- g：得到固定缓存里所有的内容并复制到模式缓存，重写其内容。
- G：得到固定缓存的内容并复制到模式缓存，添加到里面。
- I：列出不打印的字符。
- p：打印行。
- n：读下一个输入行，并开始用下一个命令而不是用第一个命令处理换行符。
- q：结束或退出 sed。

- r：从一个文件读入行。
- !：把命令应用到除选出的行以外的其他所有行。
- s：把一个字符串替换成另一个字符串。
- g：在一行上进行全局替换。
- p：打印行。
- w：把行写到一个文件中。
- x：用模式空间的内容交换固定缓存的内容。
- y：把一个字符转换成另一个（不能和正则表达式元字符一起使用）

退出状态如下。

该命令返回以下返回值。

- b：成功完成。
- >0：发生错误。

应用实例如下。

（1）执行全局更改。

```
#sed  "s/happy/enchanted/g" chap1  >chap1.new
```

该命令序列用单词 enchanted 替换每个在文件 chap1 中出现的单词 happy。它将已编辑的版本写入名为 chap1.new 的单独文件中。s 子命令结尾的 g 字符告诉 sed 命令在每行做尽可能多的替代。如果没有 g 字符，则 sed 命令只会替换行中首次出现的单词 happy。

sed 命令作为过滤器操作，它从标准输入或命令行指定的文件中（本例中是 chap1）读取、修改文本，并将其写到标准输出。与大多数的编辑器不同，它不替换原始文件，这会使 sed 命令在用于管道时成为一个强大的命令。

（2）在管道中将 sed 命令作为过滤器使用。

```
#pr  chap2 | sed "s/Page *[0-9]*$/(&)/" | enq
```

该命令序列在打印文件 chap2 之前将页码包含在圆括号中。pr 命令将标题和页码放置在每页的顶部，sed 命令将页码放在圆括号中，enq 命令打印已编辑的列表。

sed 命令模式/Page *[0-9]*$/与在行尾出现的页码相匹配。s 子命令将其更改为（&），其中&代表匹配的页码。

（3）显示文件的选定的行。

```
#sed  -n  "/food/p" chap3sed -n
```

显示文件 chap3 中包含单词 food 的每一行。通常，sed 命令将编辑过的每行复制到标准输出。-n 终止 sed 命令进行该操作，然后使用子命令（如 p）写入文本的特定部分。如果没有-n 标志，该示例会显示文件 chap3 中的所有行，并显示两次包含 food 的每个行。

（4）执行复杂的编辑。

```
#sed  -f  script.sed  chap4  >chap4.new
```

当想做任何复杂的编辑时，该命令序列创建一个 sed 脚本文件，然后可以在使用该脚本之前对其进行测试和修改。也可以重用脚本来编辑其他文件，用交互式文本编辑器创建脚本文件。

13.3 pico：编辑文字文件

作用：pico 命令用来编辑文字文件。

用法：pico [-选项] [-n<间隔秒数>] [-o<工作目录>] [-r<编辑页宽>] [-s<拼字检查器>] [+<列数编号>] [文件]

主要选项如下。

- -b：开启置换的功能。
- -d：开启删除的功能。
- -e：使用完整的文件名称。
- -f：支持键盘上的 F1、F2 等功能键。
- -g：显示光标。
- -h：在线帮助。
- -j：开启切换的功能。
- -k：预设 pico 在使用剪切命令时，会把光标所在列的内容全部删除。
- -m：开启鼠标支持的功能，可用鼠标点选命令列表。
- -n<间隔秒数>：设置多久检查一次新邮件。
- -o<工作目录>：设置工作目录。
- -q：忽略预设值。
- -r<编辑页宽>：设置编辑文件的页宽。
- -s<拼字检查器>：另外指定拼字检查器。
- -t：启动工具模式。
- -v：启动阅读模式，用户只能观看，无法编辑文件的内容。
- -w：关闭自动换行，通过这个参数可以编辑内容很长的列。
- -x：关闭界面下方的命令列表。
- -z：让 pico 可被 Ctrl+Z 组合键中断，暂存在后台作业里。
- +<列数编号>：执行 pico 指令进入编辑模式时，从指定的列数开始编辑。

说明

pico 是一个简单易用、以显示导向为主的文字编辑程序，它伴随着处理电子邮件和新闻组的程序 pine 而来。pico 很像 DOS 下的 edit 程序，打开之后，屏幕下方有操作的提示信息，功能简单，按照提示信息即可完成操作。pico 的功能键直接列在屏幕的最下方，可以作为帮助来指导操作。pico 是 pine 程序自带的，因此需要安装 pine 软件包才可以使用该编辑器。

应用实例如下。

使用 pico。

```
#pico
```

如图 13-1 所示是 pico 的工作界面。

图 13-1　pico 的工作界面

在屏幕的最上方一行显示的是系统信息，包括 pico 的版本号，以及当前正在编辑的文件名（如果正在编辑的是一个尚未保存过的新文件，则会显示 New Buffer）。如果缓冲区中右上为保存过的修改，在右上角还会出现 Modified 提示。在屏幕的最下方两行是常用的系统命令，每个命令都是一个组合键，也就是同时按下 Ctrl 键（在 pico 提示中用^符号表示按下 Ctrl 键）和表示该命令的字母。虽然在提示中用的字母都是大写，但是实际操作中并不需要输入大写字母。例如，调用系统帮助的命令是^G，我们只需同时按下 Ctrl 键和 G 键就可以了。下面列出常用的一些 pico 命令。

- Ctrl+G：获得系统帮助。
- Ctrl+O：保存文件。如果这是一个新创建的文件，则会要求用户输入一个文件名。
- Ctrl+R：要求用户输入一个文本文件的文件名，将该文件的内容插入到当前光标位置。
- Ctrl+Y：向前翻页。
- Ctrl+V：向后翻页。
- Ctrl+W：调用搜索功能。
- Ctrl+K：删除光标所在的行，并将该行的内容放入粘贴缓冲区。
- Ctrl+U：将粘贴缓冲区中的内容粘贴到当前光标位置。
- Ctrl+C：报告当前光标位置。
- Ctrl+T：调用拼写检查功能。
- Ctrl+J：段落重排功能。
- Ctrl+X：退出 pico。

13.4 vim：全屏幕文本编辑器

作用：vim 命令是命令行全屏幕文本编辑器。vim 是一个同 vi 向上兼容的文本编辑器，可以用来编辑任何 ASCII 文本，特别适合用来编写程序，它对 vi 做了许多增强：多层撤销、多窗口、多缓冲区（Buffer）、高亮度语法显示、命令行编辑、文件名匹配、在线帮助、可视选定。

用法：vim [参数] [文件...]　　编辑指定的文件
vim [参数] -　　从标准输入（stdin）读取文本
vim [参数] -t tag　　编辑 tag 定义处的文件
vim [参数] -q [errorfile]　　编辑第一个出错处的文件

主要选项如下。

- --：在这以后只有文件名。
- -v：Vi 模式（同“vi”）。
- -e：Ex 模式（同“ex”）。
- -s：安静（批处理）模式（只能与“ex”一起使用）。
- -d：Diff 模式（同“vimdiff”）。
- -y：容易模式（同“evim”，无模式）。
- -R：只读模式（同“view”）。
- -Z：限制模式（同“rvim”）。
- -m：不可修改（写入文件）。
- -M：文本不可修改。
- -b：二进制模式。
- -l：Lisp 模式。
- -C：兼容传统的 Vi: 'compatible'。
- -N：不完全兼容传统的 Vi: 'nocompatible'。
- -V[N]：Verbose 等级。
- -D：调试模式。
- -n：不使用交换文件，只使用内存。
- -r：列出交换文件并退出。
- -r（跟文件名）：恢复崩溃的会话。
- -L：同-r。
- -A：以 Arabic 模式启动。
- -H：以 Hebrew 模式启动。
- -F：以 Farsi 模式启动。
- -T<terminal>：设定终端类型为<terminal>。
- -u<vimrc>：使用<vimrc>替代任何.vimrc。
- --noplugin：不加载 plugin 脚本。
- -P[N]：打开 *N* 个标签页（默认值：每个文件一个）。

- -o[N]：打开 *N* 个窗口（默认值：每个文件一个）。
- -O[N]：同-o 但垂直分割。
- +：启动后跳到文件末尾。
- +<lnum>：启动后跳到第<lnum>行。
- --cmd <command>：加载任何.vimrc 文件前执行<command>。
- -c <command>：加载第一个文件后执行<command>。
- -S <session>：加载第一个文件后执行文件<session>。
- -s <scriptin>：从文件<scriptin>读入正常模式的命令。
- -w <scriptout>：将所有输入的命令追加到文件<scriptout>。
- -W <scriptout>：将所有输入的命令写入到文件<scriptout>。
- -x：编辑加密的文件。
- -i <viminfo>：使用<viminfo>取代.viminfo。
- -h 或--help：打印帮助信息并退出。
- --version：打印版本信息并退出。

说明

20 世纪 70 年代，人们在 UNIX 系统中使用行编辑器编辑文件时，必须发送烦琐的命令，而且无法将光标移到特定的单词纠正错误。例如，要纠正第 14 行中一个单词的拼写错误，必须发送类似 14/s/misteak/mistake/的命令。为了能将光标移到文件中的特定位置进行编辑，加利福尼亚大学伯克利分校的一名研究生 BillJoy 编写了 vi 可视化编辑器。vi 作为一个由来已久的编辑程序，不是字处理程序，不支持目前字处理程序的很多特征，它没有鼠标、下拉菜单和页面格式。每个 UNIX 系统都带有一个 vi 编辑器，它只使用很少的计算资源，但包括很多高效的编辑工具，一旦掌握了其命令，速度非常快。广泛使用的 vi 编辑器有两个版本，即标准的 UNIX 发布版包括的 vi 和 Linux 提供的 vim。vim 包括了对 vi 的一些改进。同样的命令 vi，大部分 Linux 发行版现在都带有 vim（表示 Vi IMproved）编辑器而不是经典的 vi。vi 有两种模式：命令模式（command mode）和追加/插入模式（insert/append mode），这两种模式不易掌握，需要从键盘输入的许多命令比较难记。但是一旦掌握了 vi，就会发现它作为 UNIX 系统基本工具的优势。

- 因为它是所有 UNIX 标准发布版的一部分，所以随处可用。
- 一旦掌握了基本的命令，vi 就是一个快速高效的编辑器。
- 尽管 vi 使用很少的计算资源，但是它拥有其他编辑器没有的高级特征。

1．Linux 下 vi 和 vim 命令的差别

简单来说，vi 和 vim 都是多模式编辑器，不同的是 vim 是 vi 的升级版本，它不仅兼容 vi 的所有指令，而且还包括一些新的特性。vim 的这些优势主要体现在以下几个方面。

1）多级撤销

我们知道，在 vi 里按 u 键只能撤销上一次使用的命令，而在 vim 里可以无限制地撤销。

2）易用性

vi 只能运行于 UNIX 中，而 vim 可以运行于 UNIX、Windows、Mac 等多个操作平台。

3）语法加亮

vim 可以用不同的颜色来加亮代码。比如用 putty ssh 登录 centos5.4 的时候，用 vi 不能高亮显示语法的时候，就可以用 vim 来实现了。

4）可视化操作

vim 不仅可以在终端运行，也可以运行于 X Window、Mac OS、Windows。

5）对 vi 的完全兼容

在某些情况下，可以把 vim 当成 vi 来使用。

2．安装

如果从 Linux 发布版直接安装 vim，需要注意的一点是，默认情况下系统并不一定为你安装一个完整的 vim。例如，在 Red Hat 的发布版中，vim 被拆成了 4 个包：vim-common（公用部分）、vim-minimal（最小安装）、vim-enhanced（除 X Window 支持外的完整安装）和 vim-X11（X Window 图形界面支持）。最小安装不能完整展示 vim 的优点，通常只是作为 vi 的替代品出现，缺少很多重要的特性，如多字节语言支持、鼠标支持和脚本支持。如果安装了 X Window，图形界面的 gvim 也比文本模式的 vim 具有更多的特性。建议大家尽可能安装完整的 vim。

3．vim 模式

与大部分其他编辑器不同，进入 vim 后，默认状态下输入的字符并不会插入到所编辑的文件之中。vim 的模式（mode，可以简单地理解为“状态”）概念非常重要。需要知道，vim 有以下几个模式。

- 正常（normal）模式：默认的编辑模式；下面如果不加特殊说明，提到的命令都直接在正常模式下输入；在任何其他模式中都可以通过 Esc 键回到正常模式。
- 命令（command）模式：用于执行较长、较复杂的命令；在正常模式下输入“:”（一般命令）、“/”（正向搜索）或“?”（反向搜索）即可进入该模式；命令模式下的命令要按回车键（Enter 键）才算完成。
- 插入（insert）模式：输入文本时使用；在正常模式下输入“i”（insert）或“a”（append）即可进入插入模式（也有另外一些命令，如“c”，也可以进入插入模式，但这些命令有其他的作用）。
- 可视（visual）模式：用于选定文本块；可以在正常模式下输入“v”（小写）来按字符选定，输入“V”（大写）来按行选定，或输入“Ctrl-V”来按方块选定。
- 选择（select）模式：与普通的 Windows 编辑器较为接近的选择文本块的方式；在以可视模式和选择模式之一选定文本块之后，可以输入“Ctrl-G”切换到另一模式——该模式很少在 Linux 上使用。

4．Shell 别名

如果用户在使用编辑器时习惯于输入“vi”，但是实际上却更喜欢使用 vim，那么就可以借助于别名。下面显示了如何使用 alias 命令来实现这种功能。

```
$ alias vi='vim'
$ which vi
alias vi='vim'
/usr/bin/vim
```

> **注意**
>
> 在这个例子中，如果使用 which 命令来查看 vi 程序的位置，那就会看到两行输出：第一行是别名，第二行是 vim 的位置（/usr/bin/vim）。

5．移动

以下命令可以帮助用户在文件内移动。

- h：在当前行向左移动一个字符。
- j：向下移动到下一行。
- k：向上移动到上一行。
- l：在当前行向右移动一个字符。
- w：移动到当前行下一个字。
- e：移动到当前行下一个字的结束处。
- b：移动到当前行上一个字的开始处。
- Ctrl-f：向前滚动一页。
- Ctrl-b：向后滚动一页。

如果在这些命令之前输入数字，则命令将执行该数字表示的次数，该数字称为“重复次数”或者“计数”。例如，5h 将向左移动 5 个字符。

6．移动到行

以下命令将帮助用户移动到文件的特定行。

- G：移动到文件的特定行。例如，3G 移动到第 3 行。如果没有参数，G 将移动到文件的最后一行。
- H：相对于屏幕移动到最上面的行。例如，3H 将移动到屏幕上方往下第三行。
- L：就像 H 一样，不同之处在于移动到屏幕的最后一行。因此，2L 移动到屏幕的倒数第二行。

7．离开

- :q!：退出编辑文件并放弃所有更改。这是避免麻烦的常用手段。
- :w!：写入文件（无论是否修改）。尝试重写现有文件或只读文件，或者其他不可写文件。你可以给定一个文件名作为参数，写入的将是这个文件而不是开始的文件。忽略“!”

通常会更加安全，除非你知道自己在做什么。

- ZZ：如果修改了文件，则写入文件。这是普通 vi 退出常用的方法。
- :e!：编辑文件的当前磁盘副本。这将重载文件，放弃所做的更改。
- :!：运行一个 Shell 命令。输入命令并按 Enter 键。当命令完成时，你将看到输出和提示返回到 vi 编辑。

8．修改文本

当需要插入、删除或修改文本时，可以使用以下命令。注意，有些命令具有类似于小写形式的大写形式，具体见下文的描述。

- i：在当前位置的字符之前进入插入模式。输入文本并按 Esc 键返回命令模式。使用 I 在当前行的前面插入。
- a：在当前位置的字符之后插入模型。输入文本并按 Esc 键返回到命令模式。使用 A 插入当前行的结尾。
- c：使用 c 更改当前字符并进入插入模式输入替换字符。
- o：在当前行下方另起一行以插入文本。使用 O 打开当前行上面的一行。
- cw：删除当前字的其余部分，进入插入模式进行替换。使用重复计数替换多个字。使用 c$替换行的末尾。
- dw：与上述 cw（和 c$）类似，但不进入插入模式。
- dd：删除当前行。使用重复计数删除多行。
- x：删除光标位置的字符。使用重复计数删除多个字符。
- p：将最后一个删除文本放在当前字符处。使用 P 将其放在当前字符之前。
- xp：组合 x 和 p 是一个常用的方法。这将交换光标位置的字符和它右边的字符。

9．搜索文本

可以使用表达式在文件中搜索文本。

- /：使用/后跟正则表达式可以在文件中进行向前搜索。
- ?：使用?后跟正则表达式可以在文件中进行向后搜索。
- n：使用 n 以任意方向重复上一次搜索。

应用实例如下。

1）使用 vim 编辑多个文件

vim 允许一次打开多个文件进行编辑，并在它们之间进行切换，例如：

```
# vim he.c hello.c
```

可以一次编辑两个文件。打开的屏幕会显示第一个文件，可以用底行模式下的“e”来实现两个文件的切换。

```
：e hello.c
```

也可以按 Ctrl+Shift+6 组合键回到上一个编辑的文件。注意在文件间切换时，vim 会强制保存或放弃本次修改的内容。一次编辑多个文件是很实用的功能。我们还可以在 vi 中用“:e”命令随时打开新的文件，用“yy”命令复制其中的内容，并粘贴到原来的文件中。同时编辑多

个文件时还可以用更直观的方法：分割当前屏幕，用不同的屏幕空间显示不同的文件，使编辑工作更清晰。在底行模式下使用“:sp hello.c”，即可以此种方式打开文件，并在上下两栏中分别显示。

2）在 vim 编辑器中与 Shell 交互

在 vim 中，可以在底行模式下用“！”符号来访问 Linux 的 Shell，进行操作。

“:！命令”：直接在！后面输入所要执行的命令即可。例如，在编辑/etc/service 文件时，若想知道当前主机名称，就可以在底行模式下输入：

```
：！ hostname
```

这时就会在当前屏幕中显示结果，完毕后会提示用户按 Enter 键退出。

3）加密和取消加密

（1）加密文件。在命令模式输入“:X”命令，接着输入密码、确认输入密码，最后输入“:wq”命令保存退出，文件就已经被加密了，如图 13-2 所示。

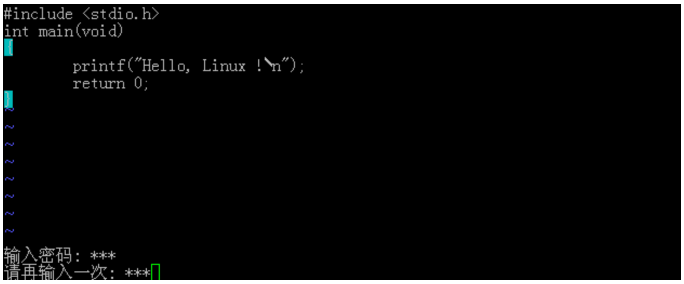


图 13-2　加密文件

经过以上方法加密后的文件变成密文文件。要阅读原来的明文，则在打开文件后会提示用户输入密码。如果密码输入正确，就能看到原来的明文。另外可以看到，在最后的状态行上有此文件已被加密的提示。

（2）取消加密。取消加密的方法很简单。在命令模式输入“:X”命令，首先会提示输入密码，这时直接按 Enter 键。接下来在提示重复输入密码时也直接按 Enter 键，最后输入“:wq”命令保存退出。这样文件的加密就被取消了。注意，以上加密方法运行在 RHEL 5.0 上的 vim 中。在其他平台可能略有差异，请以实际平台或版本的帮助为准。

4）快速查看函数帮助

在用 vim 阅读和编写程序（主要是指 C 或 C++程序）时，如果需要了解某个函数的具体功能，可以将光标移到该函数上，然后在命令模式下输入“k”，就可以调动“Linux Programmer's Manual”中对此函数的 man 帮助。

5）语法加亮

在 vim 中编写程序时，如果为了方便想开启语法加亮功能，只需在命令模式下输入“:syntax on”即可。同理可知，如果想关闭语法加亮功能，就在命令模式下输入“:syntax off”。

6）程序自动对齐

在命令模式下使用“:set ai”命令可以让程序中的各行自动对齐。

7）显示匹配

在编程中{}、[]、()常常是成对出现的，而且常常有多层的嵌套，很容易遗漏和失误，所以下面的这个技巧非常有用。在命令模式下执行“:set sm”命令会在用户输入)、]、}时显示出对应匹配的{、[、(来。

8）缩写文本

在编程中常常要输入一些重复且很长的信息，比如较长的名词和公司信息等，这时可以用 vim 的缩写功能。例如，“ad”指代“advertisement”。vim 让用户输入缩写，然后自动扩展。

用以下命令告诉 vim，每当输入“ad”时就把它扩展成“advertisement”：

```
:iabbrev ad advertisement
```

现在，当用户输入“ad”时，完整的单词“advertisement”就被插入文本。

9）二进制编辑方式

除了传统的 ASCII 码的编辑方式，vi 中还可以采用二进制的方式来编辑文件，尤其是在一些特殊目的的编程中会用到。首先以二进制方式编辑这个文件。

```
vim -b datafile
```

现在用 xxd 命令把这个文件转换成十六进制。

```
:%!xxd
```

13.5 ex：文本编辑器

作用：在 ex 模式下启动 vim 文本编辑器。ex 执行效果如同 vim -E，使用语法及参数可参照 vim 指令。

用法：和 vim 相同。

主要选项：和 vim 相同。

应用实例如下。

编辑一个文本文件。

```
#ex hello.txt
```

系统首先显示行数和字符数，如果要从 ex 模式回到普通模式，则在 vim 中输入:vi 或:visual 即可。

13.6 nano：文本编辑器

作用：nano 是类 UNXI 系统中的一个文本编辑器，是 pico 的复制品。nano 的目标是成为类似 pico 的全功能但又易于使用的编辑器。nano 是遵守 GNU 通用公共许可证的自由软件，自

从 2.0.7 版发布，许可证从 gplv2 升级到 gplv3。

用法：nano [选项] [[+行,列] 文件名]...

主要选项如表 13-1 所示。

表 13-1　主要选项及其含义

短　选　项	GNU 长选项	含　义
-h,-?	--help	显示此信息
+行,列		从所指列数与行数开始
-A	--smarthome	启用智能 Home 键
-B	--backup	存储既有文件的备份
-C<目录>	--backupdir=<目录>	用以存储独一备份文件的目录
-D	--boldtext	用粗体替代颜色反转
-E	--tabstospaces	将已输入的制表符转换为空白
-F	--multibuffer	启用多重文件缓冲区功能
-H	--historylog	记录与读取搜索/替换的历史字符串
-I	--ignorercfiles	不要参考 nanorc 文件
-K	--rebindkeypad	修正数字键区按键混淆问题
-L	--nonewlines	不要将换行加到文件末端
-N	--noconvert	不要从 DOS/Mac 格式转换
-O	--morespace	编辑时多使用一行
-P	--poslog	记录并读取光标位置
-Q<字符串>	--quotestr=<字符串>	引用代表字符串
-R	--restricted	限制模式
-S	--smooth	按行滚动而不是按半屏滚动
-T<#列数>	--tabsize=<#列数>	设定制表符宽度为“#列数”
-U	--quickblank	状态行快速闪动
-V	--version	显示版本资讯并离开
-W	--wordbounds	更正确地侦测单字边界
-Y<字符串>	--syntax=<字符串>	用于加亮的语法定义
-c	--const	持续显示游标位置
-d	--rebinddelete	修正退格键/删除键混淆问题
-i	--autoindent	自动缩进新行
-k	--cut	从游标剪切至行尾
-l	--nofollow	不要依照符号连接，而是覆盖
-m	--mouse	启用鼠标功能
-o<目录>	--operatingdir=<目录>	设定操作目录
-p	--preserve	保留 XON（^Q）和 XOFF（^S）按键
-q	--quiet	忽略软件启动问题

续表

短 选 项	GNU 长选项	含 义
-r<#列数>	--fill=<#列数>	设定行宽度为“#列数”
-s<程序>	--speller=<程序>	启用替代的拼写检查程序
-t	--tempfile	离开时自动存储，不要提示
-u	--undo	允许撤销功能
-v	--view	查看（只读）模式
-w	--nowrap	不要自动换行
-x	--nohelp	不要显示辅助区
-z	--suspend	启用暂停功能
-$	--softwrap	启用软换行
-a,-b,-e, -f,-g,-j		（忽略，为与 pico 相容）

说明

nano 像 pico 一样，是面向键盘的，它通过 Ctrl 键来控制。与 pico 不同的是，nano 使用 meta 键来启用/禁用它的功能。例如，meta-s 用来启用/禁用平滑滚动模式。几乎所有可以通过命令行开启的功能都可以像这样动态地打开或关闭。nano 同时支持用鼠标单击屏幕下方的快捷键来操作，快捷键如下。

- Ctrl+G：获得系统帮助。
- Ctrl+O：保存文件。如果这是一个新创建的文件，则会要求用户输入一个文件名。
- Ctrl+R：要求用户输入一个文本文件的文件名，将该文件的内容插入到当前光标位置。
- Ctrl+Y：向前翻页。
- Ctrl+V：向后翻页。
- Ctrl+W：调用搜索功能。
- Ctrl+K：删除光标所在的行，并将该行的内容放入粘贴缓冲区。
- Ctrl+U：将粘贴缓冲区中的内容粘贴到当前光标位置。
- Ctrl+C：报告当前光标位置。
- Ctrl+T：调用拼写检查功能。
- Ctrl+J：段落重排功能。
- Ctrl+X：退出。

应用实例如下。

（1）使用 nano。

```
# nano
```

如图 13-3 所示是 nano 的工作界面。

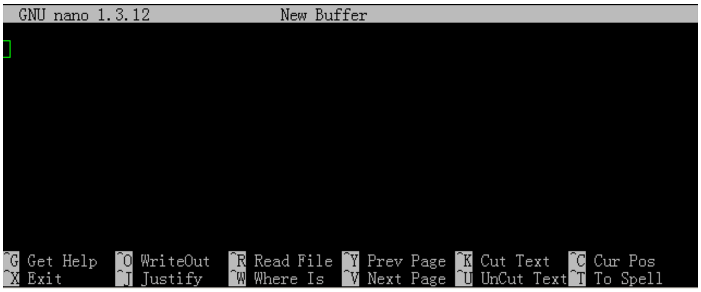


图 13-3　nano 的工作界面

在屏幕的最上方一行是系统信息，分别显示的是 nano 的版本号，以及当前正在编辑的文件名（如果正在编辑的是一个尚未保存过的新文件，则会显示 New Buffer）。如果缓冲区中右上为保存过的修改，在右上角还会出现 Modified 提示。屏幕的最下方两行是常用的系统命令。每个命令都是一个组合键，也就是同时按下 Ctrl 键（在 pico 提示中用^符号表示按下 Ctrl 键）和表示该命令的字母。虽然提示用的字母都是大写，但是实际操作中并不需要输入大写字母。例如，调用系统帮助的命令是 Ctrl+G，我们只需要同时按下 Ctrl 键和 G 键就可以了。

（2）禁用自动换行。

```
# nano -w /etc/fstab
```

警告：这非常重要。如果在编辑/etc/fstab 配置文件时忘了加-w 参数，可能会导致系统无法启动或产生其他异常。

13.7　jed：编辑文本文件

作用：Jed 是以 Slang 所编写的程序，适合用来编辑程序原始代码。

用法：jed [-2n] [-batch] [-f<函数>] [-g<行数>] [-i<文件>] [-I<文件>] [-s<字符串>] [文件]

主要选项如下。

- -2：显示上下两个编辑区。
- -batch：以批处理模式来执行。
- -f<函数>：执行 Slang 函数。
- -g<行数>：移到缓冲区中指定的行数。
- -i<文件>：将指定的文件载入缓冲区。
- -i<文件>：载入 Slang 原始代码文件。
- -n：不要载入 jed.rc 配置文件。
- -s<字符串>：查找并移到指定的字符串。

应用实例如下。

使用两个编辑区编辑 hello.c 文件。

```
#jod -2 hello.c
```

13.8 joe：编辑文本文件

作用：joe 是一个功能强大的全屏幕文本编辑程序，操作的复杂度要比 pico 高一点，但是功能较为齐全。joe 一次可开启多个文件，每个文件各放在一个编辑区内，并可在文件之间执行剪贴的动作。

用法：joe [选项] [要编辑的文件]

主要选项如下。

- -asis：字符码超过 127 的字符不做任何处理。
- -backpath<目录>：指定备份文件的目录。
- -beep：编辑时，若有错误即发出哗声。
- -columns<栏位>：设置栏数。
- -csmode：可执行连续查找模式。
- -dopadding：在程序与 tty 之间存在缓冲区。
- -exask：在程序中，执行“Ctrl+K+X”组合键时，会先确认是否要保存文件。
- -force：强制在最后一行的结尾处加上换行符号。
- -help：执行程序时一并显示帮助。
- -keepup：在进入程序后，画面上方为状态列。
- -lightof：选取的区块在执行完区块命令后，就会恢复成原来的状态。
- -lines<行数>：设置行数。
- -marking：在选取区块时，反白区块会随着光标移动。
- -mid：当光标移出画面时，即自动卷页，使光标回到中央。
- -nobackups：不建立备份文件。
- -nonotice：程序执行时，不显示版权信息。
- -nosta：程序执行时，不显示状态列。
- -noxon：尝试取消 Ctrl+S 与 Ctrl+Q 组合键的功能。
- -orphan：若同时开启一个以上的文件，则其他文件会置于独立的缓冲区，而不会另外开启编辑区。
- -pg<行数>：按 PageUp 或 PageDown 键换页时，所要保留前一页的行数。
- -skiptop<行数>：不使用屏幕上方指定的行数。
- +<行数>：指定开启文件时，光标所在的行数。
- -autoindent：自动缩排。
- -crlf：在换行时，使用 CR-LF 字符。
- -indentc<缩排字符>：执行缩排时，实际插入的字符。
- -istep<缩排字符数>：每次执行缩排时，所移动的缩排字符数。
- -keymap<按键配置文件>：使用不同的按键配置文件。
- -linums：在每行前面加上行号。
- -lmargin<栏数>：设置左侧边界。

– -overwrite：设置覆盖模式。
– -rmargin<栏数>：设置右侧边界。
– -tab<栏数>：设置 Tab 字符的宽度。
– -rdonly：以只读的方式开启文件-wordwrap 编辑时若超过右侧边界，则自动换行。

应用实例如下。

打开 hello.c 文件第二行进行编辑。

```
#joe +2 hello.c
```

14 Linux Shell 命令

14.1 Shell 简介

14.1.1 什么是 Shell

Shell 的中文含义是“外壳”，通俗地讲，Shell 是一个交互编程接口，通过获得用户输入来驱动操作系统内核完成指定工作。它本身在用户与操作系统之间起到桥梁的作用。Shell 脚本在本质上是一种解释型语言。Shell 除作为命令解释程序以外，还是一种高级程序设计语言，它有变量、关键字，有各种控制语句、支持函数模块，有自己的语法结构，利用 Shell 程序设计语言可以编写出功能很强但代码简单的程序。Shell 是系统核心的一个重要外延，它能够调用系统内核的大部分功能。因此可以说，Shell 是 UNIX/Linux 最重要的实用程序。Shell 和内核用户的关系可以用图 14-1 表示。

图 14-2 描述了 Shell 是如何完成命令解释的。

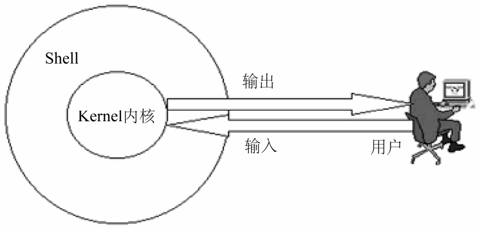


图 14-1　Shell 和内核用户的关系

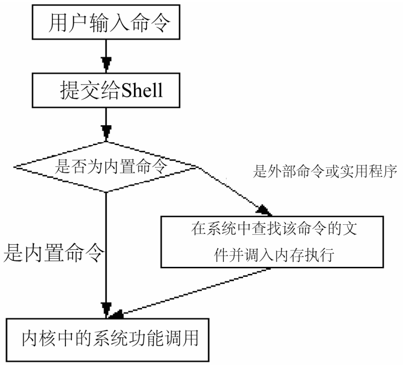


图 14-2　命令解释过程

Shell 最重要的功能是命令解释，从这种意义上说，Shell 是一个命令解释器。Linux 系统上可执行文件的分类如下。

- Linux 命令：存放在/bin、/sbin 目录下的命令。
- 内置命令：出于效率的考虑，将一些常用命令的解释程序构造在 Shell 内部。
- 实用程序：存放在/usr/bin、/usr/sbin、/usr/local/bin 等目录下的实用程序或工具。

– 用户程序：用户程序经过编译生成可执行文件后，也可作为 Shell 命令运行。
– Shell 脚本：由 Shell 语言编写的批处理文件。

14.1.2　Shell 类型

Linux 系统提供多种不同的 Shell 以供选择，常用的有 Bourne Shell（sh）、C-Shell（csh）、Korn Shell（ksh）和 Bourne Again Shell（bash）。

Bourne Shell 是 AT&T Bell 实验室的 Steven Bourne 为 AT&T 的 UNIX 开发的，它是 UNIX 的默认 Shell，也是其他 Shell 的开发基础。

C Shell 是加利福尼亚大学伯克利分校的 Bill Joy 为 BSD UNIX 开发的，与 sh 不同，它的语法与 C 语言很相似。它是由 Bill Joe 于 1983 年写成的，由标准的 BSD（Berkeley System Distribution）命令解释。它提供了 Bourne Shell 所不能处理的用户交互特征，如命令补全、命令别名、历史命令替换等。

有很长一段时间，只有两类 Shell 供人们选择：Bourne Shell 用来编程，C Shell 用来交互。为了改变这种状况，AT&T 的 Bell 实验室 David Korn 开发了 Korn Shell。它集合了 C Shell 和 Bourne Shell 的优点，并且与 Bourne Shell 向下完全兼容。

Bourne Again Shell（即 Bash）是自由软件基金会（GNU）开发的一个 Shell，它是 Linux 系统中一个默认的 Shell。Bash 不但与 Bourne Shell 兼容，还继承了 C Shell 和 Korn Shell 等的优点。Linux 使用它作为默认的 Shell 是因为它具有以下特色。

– 可以使用类似 DOS 下面的 doskey 的功能，用上下方向键查阅和快速输入并修改命令。
– 自动通过查找匹配的方式，给出以某字串开头的命令。
– 包含了自身的帮助功能，只要在提示符下输入 help，就可以得到相关的帮助。

其他 Shell 如下。

– zch 是 Linux 最大的 Shell 之一，由 Paul Falstad 完成，共有 84 个内部命令。如果只是一般的用途，是没有必要安装这样的 Shell 的。
– csh 是 Linux 比较大的内核，它由以 William Joy 为代表的共计 47 位作者编成，共有 52 个内部命令。该 Shell 其实是指向/bin/tcsh 这样的一个 Shell，也就是说，csh 其实就是 tcsh。
– POSIX Shell 是 Korn Shell 的一个变种。当前提供 POSIX Shell 的最大卖主是 Hewlett-Packard。在 HP-UX 11.0 中，POSIX Shell 就是/bin/sh，而 bsh 是/usr/old/bin/sh。

在过去的 30 年中，UNIX Shell 获得了许多改进和增强，结果出现了多种不同的 Shell。图 14-3 显示了部分 UNIX Shell 的家族树。虽然该图并不完整，但它显示了衍生出其他较次要 Shell 的那些主要 Shell。

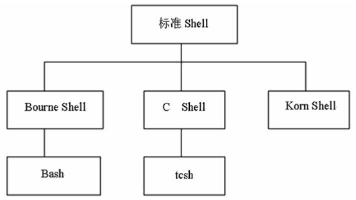


图 14-3　部分 UNIX Shell 的家族树

14.2 Linux Shell 的功能

Shell 最重要的功能是命令解释，从这种意义上说，Shell 是一个命令解释器。但是除命令外，Shell 还具有如下一些重要功能。

14.2.1 通配符

可能你已经在不知不觉中使用了通配符。当你在指定文件或寻找文件时，使用*代表任意字符串。例如，*.txt 同所有以.txt 结尾的文件相匹配。不过，除了*，还有很多其他的通配符。当输入 ls *.txt 命令并按 Enter 键后，寻找哪些文件同*.txt 模式相匹配的任务不是通过 ls 命令来完成的，而是由 Shell 自己完成的。这需要对命令行是如何被 Shell 解析的做进一步解释。输入如下代码：

```
$ ls *.txt
    readme.txt  recipes.txt
```

该命令首先被分割成一系列单词（本例中的 ls 和*.txt）。如果 Shell 在某个单词中发现了*，它会将整个单词当作通配模式解析，并用所有相匹配的文件替换它。因此，该命令在执行前就变为 ls readme.txt recipe.txt，而这将得到期望的结果。其余通配符如下。

- ?：仅与一个任意字符匹配。
- [...]：同方括号中的任意一个字符相匹配。这些字符可以用字符范围（比如 1～9）、离散值或同时使用两者表示。例如，[a-zBE5-7]同所有 a～z 之间的字符和 B、E、5、6、7 相匹配。
- [!...]：与所有不在方括号中的某个字符匹配。例如[!a-z]同某个非小写字母相匹配。
- {c1,c2}：同 c1 或者 c2 相匹配。其中 c1 和 c2 也是通配符。因此，可以使用{[0-9]*,[acr]}。

以下是一些通配符模式及其说明。

- /etc/*conf：/etc 目录中所有以 conf 结尾的文件。它将同/etc/inetd.conf、/etc/conf.linuxconf 相匹配，并且也会同/etc/conf 相匹配。请注意，*也匹配空字符串。
- image/{cars,space[0-9]}/*.jpg：image/cars、image/space0、(...)、image/space9 目录中以.jpg 结尾的文件。
- /usr/share/doc/*/README：所有/usr/share/doc 的直接子目录中的全部 README 文件，比如/usr/share/doc/mandrake/README，但是不包括/usr/share/doc/myprog/ doc/README。
- *[!a-z]：当前目录中不以小写字符结尾的全部文件。

14.2.2 重定向

每个命令都使用标准输入流、标准输出流和标准错误流。这些标准流被预告分配给键盘和显示器。在需要的时候，可以使用重定向临时改变这些默认分配。重定向是这样一种过程，我们借助它指定一个用于替代某个标准文件的文件。对于输入文件，称之为输入重定向；对于输

出文件，称之为输出重定向；而对于错误文件，称之为错误重定向。

当 Shell 运行一个新的应用程序或命令时，它会为进程自动创建 3 个文件描述符，即标准输入（stdin）、标准输出（stdout）和标准错误输出（stderr），它们被分别赋值为 0、1 和 2。标准输入通常默认为终端的键盘，标准输出和标准错误输出通常默认为终端的屏幕。在默认情况下，应用程序会从 stdin 中取得输入数据，并把输出数据和出错信息分别传送给 stdout 和 stderr。重定向就是要改变这些 stdin、stdout 及 stderr 默认的去向。

1．输出重定向

简单地说，输出重定向就是把应用程序的标准输出或标准错误输出重定向到某个指定的文件，而不是原来默认的标准输出，比如屏幕。它的操作符是“>”或“>>”。运用输出重定向可以比较方便地生成、编辑某些文件，比较容易地和系统中的某些设备文件进行交互。下面是输出重定向的一些实例。

列出 home 目录下所有的 csv 文件，并把结果保存到文件 csv.out 中。

```
# find /home -name "*csv" > csv.out
```

获取系统内存使用信息并追加到文件 sysinfo.out 中。

```
# cat /proc/meminfo >> sysinfo.out
```

如果不想让别人在网络上 ping 到自己的机器，可以像下面这样做。

```
# echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

如果不想看到某个命令执行的结果，可以像下面这样将其丢弃。

```
# ls -aR /home/wei  > /dev/null
```

2．输入重定向

输入重定向是把应用程序的标准输入重定向到指定的文件，这就可以使程序的输入源不限于键盘，也可以来自文件。输入重定向的操作符是“<”或“<<”。与输出重定向相比，输入重定向运用的地方要少一些。不过在某些情况下使用它们可以起到很好的效果。下面是输入重定向的一些实例。

在当前目录下快速创建一个空文档 ra.dat。

```
# cat > ra.dat < /dev/null
```

在局域网上给其他人发送信息。

```
# mail -s "The experiem results"  user1 user2  < ra.dat
```

统计文件 test.out 中的行数和字数。

```
# wc -l -w < test.out
```

另外，Bash 所支持的重定向操作符如下。

- <filename：将标准输入重定向为文件 filename。
- >filename：除非文件 filename 已存在并且设置了 noclobber 标记，否则标准输出将被重定向到文件 filename；如果文件 filename 不存在且没有设置 noclobber 标记，那么重定向操作将创建该文件。
- >| filename：即使文件 filename 存在且设置了 noclobber 标记，仍将标准输出重定向到该文件。
- >> filename：除非文件 filename 已存在并且设置了 noclobber 标记，否则标准输出将被重定向到文件 filename，并将内容添加到原文件的末尾；如果没有设置 noclobber 标记，并且文件 filename 不存在，那么将创建该文件。
- <&m：从文件描述符 m 复制标准输入。
- [n]>&m：从文件描述符 m 复制标准输出或者文件描述符 n（如果命令中指定了 n）。
- [n]<&-：关闭标准输入或者文件描述符 n（如果指定了 n）。
- [n]>&-：关闭标准输出或者文件描述符 n（如果指定了 n）。

14.2.3 管道

经常需要使用一系列命令完成某项任务。例如，如果需要查看登录系统的用户的一个列表，则可以使用 who 命令。然而，如果需要该列表的一个硬复制，则需要两个命令。首先使用 who 命令得到该列表，并且使用重定向将结果保存在一个文件中。然后使用 lpr 命令打印该文件。通过使用管道，可以避免创建此中间文件。管道是一种运算符，它将一个命令的输出临时保存在缓冲区中，与此同时，保存于缓冲区的该输出又被用作下一个命令的输入。第一个命令必须能够发送其输出至标准输出；第二个命令必须能够从标准输入读入其输入。管道符号是一个垂直竖线（|）。键盘上不存在该竖线字符的标准位置，通常可以在键盘的右侧找到它，经常位于 Ender 键的上方。看一个例子：

```
$who | lpr
```

管道在某种程度上是输入和输出重定向的结合。其原理同物理管道类似：一个进程向管道的一端发送数据，而另一个进程从该管道的另一端读取数据。管道符是“|”。让我们再来看看上述文件列表的例子。假设想直接找出有多少对应的文件，而不想先将它们保存到一个临时文件，可以使用下面的命令：

```
#ls images/*.png | wc -l
```

这表示将 ls 命令的标准输出（即文件列表）重定向到 wc 命令的输入，这样就直接得到了想要的结果。

也可以使用下述命令得到“除去扩展名”的文件列表。

```
#ls images/*.png | sed -e 's/\.png$//g' >the_list
```

或者，如果想要直接查看结果而不想保存到某个文件，可以使用如下命令。

```
#ls images/*.png | sed -e 's/\.png$//g' | less
```

14.2.4　别名

通过将一个名称分配给命令，别名提供了一种创建定制命令的方法。每种 Shell 处理别名的方式不同，这里将分别予以讨论。在 Korn Shell 和 Bash Shell 中，别名是通过使用 alias 命令创建的。其格式为 alias name=command-definition。

其中，alias 是命令关键字，name 是要创建的别名的名称，而 command-definition 是代码。在赋值运算符（=）的前后不能包含空格。

14.2.5　命令行补全

命令行补全是一个十分有用的功能。所有现代的 Shell（包括 Bash）都有该功能。它的作用是让用户尽量少操劳。实例是解释补全的最好方式。

假设想查看个人目录中的一个叫作 file_with_very_long_name_impossible_to_type 的文件，并且假定在同一个目录中还有一个叫作 file_text 的文件。现在在你的个人目录中只要输入：

```
$ less fi<TAB>
```

即输入 less fi 然后按下 Tab 键，Shell 就会为你扩展命令行：

```
$ less file_
```

并给出可能的选项列表（这个默认配置可以自定义）。然后输入：

```
less file_w<TAB>
```

Shell 就会将命令行扩展成你所需要的：

```
less file_with_very_long_name_impossible_to_type
```

然后，你要做的就是按下 Enter 键确认，并阅读文件。

其他补全方式：虽然最为常用，但 Tab 键并非激活补全的唯一方式。一般来说，对于命令行中的第一个单词，要补全的是命令名（比如 nsl<TAB>会给出 nslookup）；对于其他单词会扩展成文件名，除非该单词以～、@或$等“特殊”字符为前缀，而这时 Shell 会分别尝试用用户名、机器名或环境变量名来进行扩展。还有一个特殊字符用于补全文件名（/）和一个用于从历史记录中回用一个命令的特殊字符（!）。另外两个激活补全的方式是依次按下 Esc-<x>和 Ctrl+x <x>组合键，其中<x>是上述特殊字符。Esc-<x>将尝试给出一个唯一的补全。如果不行的话，将给出最大可能子字符串选择列表。“哔”的一声表示要么选择不唯一，要么没有对应的选项。Ctrl+x <x>序列将不补全而显示所有可能的选项列表。按下 Tab 键相当于依次按下 Esc-<x>和 Ctrl+x <x>组合键，其中的特殊字符依赖于上下文环境。因此，要查看全部已定义的环境变量，一种方式是在空白行上依次按下 Ctrl+x $键。

14.2.6　命令替换

命令替换的用途是将命令的输出结果赋给一个变量，或将命令的输出结果代入字符串。所

有的 Shell 都使用反引号来执行命令替换。bash 允许使用两种格式：在旧的格式中，命令被放在反引号内；而在新的 Korn 风格的格式中，命令被放在前面有个美元符号的一对括号内。执行扩展时，bash 先执行命令，然后返回命令的标准输出，输出结果末尾的换行符都将被删除。使用旧风格的反引号格式进行替换时，反斜杠将保留它的字面含义，除非它后面跟的是$、`或\。当使用$（command）格式时，括号内的所有字符只用于构成命令，都不会被特殊对待。命令替换可以被嵌套。当用旧格式嵌套时，内部的反引号必须用反斜杠来转义。

格式：

```
'UNIX command'   # Old method with backquotes
$(UNIX command)  # New method
```

下面是几个例子。

```
 $ echo "The hour is `date +%H`"
   The hour is 09
```

date 命令的输出被替换到串中。

```
$ name=`awk –F: '{print $1}' database`
   $ echo $name
   Ebenezer Scrooge
```

将 awk 命令的输出赋给变量 name，并显示出来。

```
$ "  ls 'ls /etc'
   shutdown
```

反引号里那条 ls 命令的输出是/etc 目录下的文件列表，这些文件名将成为第一个 ls 命令的参数。当前目录下在/etc 里同名的文件都被列了出来。

```
 $ set `date`
```

set 命令把 date 命令的输出赋给位置参量。空白符拆分词表，并将各个词放到对应的参数中。

14.2.7　命令历史

1．命令历史简介

Shell 提供一个由命令、环境变量和文件结合而成的大量命令历史记录的功能。此设计的主要特征是具有不用再输入命令就可以重新调用命令和重新执行的功能。Shell 提供命令行历史机制。它将在命令行输入的命令保存为一个带编号的清单。在一个登录会话中，输入的命令会保存在 Shell 内存中的历史清单中，并在退出时添加到历史文件中。可以从历史列表中调出某条命令再次执行，而不必重新输入这条命令。

2．命令历史变量

命令历史变量如下。

– FCEDIT：使用 fc 命令的 UNIX/Linux 编辑器的路径名。

- HISTCMD：当前命令的历史编号，或在历史清单中的序号。若未设置，它的特性将丢失，即使之后进行重置也将不起作用。
- HISTCONTROL：如果设置了 ignorespace 的值，以空格开头的行将不会进入历史清单。如果设置了 ignoredups 的值，与最后一个历史行匹配的行不会进入。ignoreboth 的值结合了两个选项。如果没有设置，或设成了上面两个值以外的其他值，那么解释器读到的所有行都将保存在历史清单中。
- HISTFILE：指定保存命令行历史的文件。默认是～/.bash_history。如果没有设置，当一个交互式 Shell 退出时不会保存命令行历史。
- HISTFILESIZE：历史文件能包含的最大行数。当给这个变量赋值后，如果有必要，历史文件将被截尾，以使包含的行数不超过这个数。默认值是 500。
- HISTIGNORE：以冒号分隔的一系列模式，用来决定哪些命令行应该保存在历史清单中。每个模式确定行的开始且包括正常的 Shell 匹配字符模式。用于模式中的&使 history 命令忽略重复。例如，ty??:&将匹配任何以 ty 开头后跟两个字符的命令行，以及该命令的重复。这些命令不会放入历史清单中。
- HISTSIZE：记录在命令历史中的命令数。默认值是 500。

3．方向键

从历史文件中访问命令，可以用方向键在历史文件中上、下、左、右移动。可以在历史文件中编辑任意行，用标准键来删除、修改、退格等。编辑完一行后，只要按下 Enter 键就会执行该命令行。方向键操作如下。

↑：向上箭头使光标在历史命令中向上移动。

↓：向下箭头使光标在历史命令中向下移动。

→：向右箭头使光标在历史命令中向右移动。

←：向左箭头使光标在历史命令中向左移动。

4．相关命令 fc、history、bang

1）fc 命令

fc 命令也称 fix 命令，可以用于两个方面：从历史清单中选择命令；用 vi 或 Emacs 编辑器或系统上的任何其他编辑器来编辑命令。第一种格式，带-l 选项的 fc 命令可以从历史清单中选择指定的行或行的范围。当-l 选项打开时，输出发送到屏幕上。

fc 命令格式：fc [选项]

选项如下。

- -r：开关反转命令的序号。
- -e editor：将历史清单调入编辑器。
- -l n-m：列出编号从 *n* 到 *m* 的命令。
- -n：关闭历史清单的编号。
- -r：反转历史清单的序号。
- -s string：访问以 string 开头的命令。

应用实例如下。

（1）列出历史清单中的最后两条命令。

```
# fc  -l
983     emacs h.log
984     emacs h.log
```

（2）打印出没有行号的历史清单。

```
# fc -ln
        emacs h
        emacs h
        fc -help
        man fc
        ls
```

（3）显示历史清单中的最后 5 条命令，不带行号，重定向到文件 goodcjhsaved。

```
# fc -ln -5 > goodcjhsaved
```

2）history 命令

history 命令的用法请参考第 7 章。下面是应用实例。

（1）显示历史清单，将结果重定向到文件 goodcjh。

```
# history > goodcjh
```

（2）显示文件 goodcjh 的内容。

```
# more goodcjh1
```

3）bang 命令

重新执行历史清单中的命令需要使用感叹号（称为 bang）。如果输入两个感叹号（!!），即“bang”、“bang”，则历史清单中的最后一条命令将重新执行。如果输入一个感叹号，后跟一个数字，那么以该数字列出的命令将重新执行。如果输入一个感叹号和一个字母或字符串，那么最近一条以该字母或字符串开头的命令将重新执行。脱字符（^）也可以用来作为编辑过去命令的快捷方式。下面是历史替换字符。

– !：开始历史替换。
– !!：重新执行上一条命令。
– !N：重新执行历史清单中的第 *N* 条命令。
– !-N：重新执行从当前命令往回数的第 *N* 条命令。
– !string：重新执行最后一条以串 string 开头的命令。
– !?string?：重新执行最后一条包含串 string 的命令。
– !?string?%：重新执行历史清单中最近一条包含串 string 的命令行参数。
– !$：用上一条命令的最后一个参数作为当前命令行。
– !! string：将 string 添加到上一条命令的最后并执行。

– !N string：将 string 添加到历史清单中第 *N* 条命令的最后并执行。
– !N:s/old/new/：在前面的第 *N* 条命令中，将第一次出现的 old 串替换成 new 串。
– !N:gs/old/new/：在前面的第 *N* 条命令中，将所有的 old 串替换成 new 串。

应用实例如下。

（1）从历史清单中取出最后那条命令，然后再次执行该命令。

```
# !!
pwd
/root
```

（2）执行历史清单中的第 25 条命令。

```
$ !25
date
Mon Jul 12 12:29:26 PST 2008
```

14.3 Linux Shell 元字符

在 Shell 中有一些具有特殊意义的字符，称为 Shell 元字符（Shell Metacharacters）。若不以特殊方式指明，Shell 并不会把它们当作普通文字符使用。表 14-1 简单介绍了常用的 Shell 元字符及其含义。

表 14-1　常用的 Shell 元字符及其含义

元 字 符	含　义
\	按字面含义解释它后面那个字符
&	后台执行字符。在一个命令之后加上字符“&”，该命令就会以后台方式执行
;	分隔命令
$	替换变量
?	匹配单个字符
[abc]	匹配这组字符中的一个。例如 a、b 或者 c
[!abc]	匹配这组字符以外的某个字符。例如，除 a、b 或者 c 以外的字符
*	匹配 0 个或多个字符
( )	在子 Shell 中执行命令
{ }	在当前 Shell 中执行命令
\<Enter>	续行符。可以使用续行符将一个命令行分写在多行上
|	管道字符
"	在"…"中间的字符会被当作文字处理并允许变量值置换
'	在'…'中间的字符都会被当作文字处理，指令、文件名、保留字等都不再具有原来的意义

续表

元 字 符	含　　义
>	输出重定向字符
<	输入重定向字符
(cmds)	在子 Shell 中执行命令
{cmds}	在当前 Shell 中执行命令

14.4 Shell 变量和 Shell 环境

14.4.1　变量简介

变量是一个可以保存值的内存地址。每种 Shell 都允许创建、保存和访问位于变量中的值。每个 Shell 变量都必须拥有一个名称。变量名必须以一个英语字母或下画线字符开头，后接 0 个或多个字母、数字或下画线字符。存在两大变量类型：用户定义变量和预定义变量，如图 14-4 所示。

图 14-4　两大变量类型

1．用户定义变量

在 UNIX/Linux 中，用户变量并非专门定义的。对变量的第一次引用将创建它。在 Korn Shell 和 Bash Shell 中，将值保存在变量中的语法相同，但是，在 C Shell 中则不同。

2．预定义变量

预定义变量用于配置一个用户的 Shell 环境。例如，一种系统变量确定哪种编辑器用于编辑命令历史。其他系统变量则保存关于环境的信息，环境变量由系统统一命名。部分系统变量的值由系统设定，部分环境变量的值可以由用户给定。预定义变量又可以分成两类：Shell 变量和环境变量。Shell 变量用于定制 Shell。环境变量用于控制用户环境，并且可以导出至子 Shell。

3．Shell 变量

如表 14-2 所示是一个各种 Shell 中的 Shell 内置变量表。Korn Shell 和 Bash Shell 只使用大写标识符。C Shell 大写和小写都用。

表 14-2　Shell 变量及其含义

Shell 变量	含　　义
CDPATH	当目录参数是一个相对路径名时，包含 cd 命令的搜索路径
EDITOR	命令行编辑器的路径名，它既是 Shell 变量，也是环境变量
EVN	环境文件的路径名
HOME	用户根目录的路径名
PATH	命令的搜索路径

续表

Shell 变量	含　　义
PS1	提示符 1，主提示符（默认为“\s-\v\$”）
PS2	提示符 2，辅助提示符（默认为“>”）
PS3	select 发出的提示符
PS4	Bash 调试符
SHELL	登录 Shell 的路径名
TERM	终端类型
INPUTRC	Readline 初始化文件的路径名（默认为～/.inputrc）
HISTFILESIZE	保存在 HISTFILE 中的最大项数
HISTFILE	保存历史列表文件的路径名（默认为～/.bash_history）
HISTSIZE	保存在历史列表中的最大项数
FCEDIT	fc 默认使用的编辑器名称
MAILCHECK	以秒为单位定义了 Bash 检查邮件的频度
MAIL	保存用户邮件的文件的路径名
MAILPATH	Bash 检查邮件文件的路径名列表，名字之间用冒号隔开
IFS	内部字段分隔符，用于分词
BASH_ENV	用于非交互式 Shell 的初始化文件的路径名
REPLY	保存 read 接受的行，还用于 select

说明

Shell 变量为小写；环境变量为大写。

14.4.2　了解 Shell 的内置命令

Shell 有很多内置在其源代码中的命令。这些命令是内置的，所以 Shell 不必到磁盘上搜索它们，执行速度因此加快。bash 提供的 help 功能能提供任何内置命令的在线帮助。表 14-3 列出了 bash 内置命令及其功能。表 14-4 列出了 tcsh 内置命令及其功能。

表 14-3　bash 内置命令

命　　令	功　　能
.	执行当前进程环境中的程序。同 source
file	dot 命令从文件 file 中读取命令并执行
:	空操作，返回退出状态 0
alias	显示和创建已有命令的别名
bg	把作业放到后台
bind	显示当前关键字与函数的绑定情况，或者将关键字与 readline 函数或宏进行绑定

续表

命　　令	功　　能
break	从最内层循环跳出
builtin [sh-builtin [args]]	运行一个内置 Shell 命令，并传送参数，返回退出状态 0。如果一个函数与一个内置命令同名时，该命令将很有用
cd [arg]	改变目录。如果不带参数，则回到主目录；带参数，则切换到参数所指的目录
command comand [arg]	即使有同名函数，仍然执行该命令。也就是说，跳过函数查找
declare [var]	显示所有变量，或用可选属性声明变量
dirs	显示当前记录的目录（pushd 的结果）
disown	从作业表中删除一个活动作业
echo [args]	显示 args 并换行
enable	启用或禁用 Shell 内置的命令
eval [args]	把 args 读入 Shell，并执行产生的命令
exec command	运行命令，替换掉当前 Shell
exit [n]	以状态 *n* 退出 Shell
export [var]	使变量可被子 Shell 识别
fc	历史的修改命令，用于编辑历史命令
fg	把后台作业放到前台
getopts	解析并处理命令行选项
hash	控制用于加速命令查找的内部哈希表
help [command]	显示关于内置命令的有用信息。如果指定了一个命令，则将显示该命令的详细信息
history	显示带行号的命令历史列表
jobs	显示放到后台的作业
kill [-signal process]	向由 PID 或作业号指定的进程发送信号。输入 kill –l 查看信号列表
let	用来计算算术表达式的值，并把算术运算的结果赋给变量
local	用在函数中，把变量的作用域限制在函数内部
logout	退出登录 Shell
popd	从目录栈中删除项
pushd	向目录栈中增加项
pwd	打印出当前的工作目录
read [var]	从标准输入读取一行，保存到变量 var 中
readonly [var]	将变量 var 设为只读，不允许重置该变量
return [n]	从函数中退出，*n* 是指定给 return 命令的退出状态值
set	设置选项和位置参量
shift [n]	将位置参量左移 *n* 次
stop pid	暂停第 pid 号进程的运行
suspend	终止当前 Shell 的运行（对登录 Shell 无效）

续表

命　　令	功　　能
test	检查文件类型，并计算条件表达式
times	显示由当前 Shell 启动的进程运行所累计的用户时间和系统时间
trap [arg] [n]	当 Shell 收到信号 *n*（*n* 为 0、1、2 或 15）时，执行 arg
type [command]	显示命令的类型，例如 pwd 是 Shell 的一个内置命令
typeset	同 declare，设置变量并赋予其属性
ulimit	显示或设置进程可用资源的最大限额
umask [八进制数字]	用户文件关于属主、属组和其他用户的创建模式掩码
unalias	取消所有的命令别名设置
unset [name]	取消指定变量的值或函数的定义
wait [pid#n]	等待 PID 为 *n* 的后台进程结束，并报告它的结束状态

表 14-4　tcsh 内置命令

内置命令	功　　能
%job	与 fg 内置命令相同，job 为用户希望放置到前台运行的作业的编号
%job&	与 bg 内置命令相同，job 为用户希望放置到后台运行的作业的编号
@	类似于内置命令 set，但是能够计算算术表达式
alias	创建和显示别名，bash 使用与 tcsh 不同的语法
alloc	报告空闲内存总量和已用内存空间总量
bg	将一个被挂起的作业放置到后台执行
bindkey	控制按键到 tcsh 命令行编辑器命令的映射
bindkey	如果不带任何参数，bindkey 将显示所有的按键绑定
bindkey -l	列出所有可用的编辑器命令及其简短描述
bindkey -e	将命令行编辑器切换到 Emacs 模式
bindkey -v	将命令行编辑器切换到 vim 模式
bindkey key command	将编辑器命令 command 绑定到按键 key 上
bindkey -bkey command	与前一种形式类似，但是它允许用户通过 C-x 形式（其中 x 表示用户在按下 Ctrl 键时输入的字符）指定控制键，通过 M-x 指定 mata 键（在大多数 Linux 使用的键盘上这个 meta 键就是 Alt 键）序列，并通过 F-x 指定功能键
bindkey -ckey command	将按键 key 绑定到命令 command 上，这里 command 不是某个编辑器命令，而是某个 Shell 内置命令或者某个可执行程序
bindkey -skey string	每当输入 key 时，替换成 string
builtins	显示所有内置命令的列表
cd 或者 chdir	改变工作目录
dirs	显示目录栈

续表

内置命令	功　　能
echo	显示传递给该命令的参数。使用-n 选项就可以阻止 echo 在行末显示 RETURN 符。echo 内置命令类似于 echo 工具
eval	扫描并计算命令行；如果在命令行的起始处放置 eval，那么 Shell 在执行这条命令行之前，扫描该命令行两次；这个特性对于那些由命令替换或者变量替换产生的命令来说非常有用；因为 Shell 按照规定的顺序处理命令行，所以有时候需要重复扫描才能取得预期的结果
exec	在同一个 Shell 中使用另一个程序来覆盖正在执行的程序；原来的程序将被丢弃
exit	退出 TC Shell；如果这个命令后面跟着一个数值参数，那么将把这个数字作为退出状态返回
fg	将某个作业放置到前台执行
filetest	取出后面跟着一个或者多个文件名的某个文件查询操作符，并将该操作符应用到每个文件名上；返回结果为空格符分开的一个列表
glob	类似于 echo，但是在它的参数之间不显示空格符，而且在显示末尾不跟换行符
hashstat	报告 tcsh 的 hash 机制的效率，hash 机制加速了在搜索路径中的查找过程，还可以参见 rehash 和 unhash
history	显示最近使用的命令的列表
jobs	显示作业列表（被挂起的作业和那些正在后台执行的作业）
kill	终止某个作业或者进程
limit	限制当前进程及它所创建进程使用的计算机资源；用户可以限制该进程使用的秒数（CPU 时间），以及创建的文件大小等
log	立即产生平时 Shell 变量 watch 要每隔 10 分钟才能产生的报告
login	使某个用户登录，后面可以跟着用户名
logout	如果用户正在使用原 Shell（登录 Shell），那么这将结束会话
ls-F	与 ls –F 类似，但是速度更快；注意内置命令字符串 ls-F 中并不含空格符
nice	降低命令或者 Shell 的处理优先级；如果用户希望运行一条需要大量系统资源并且不想立即得到结果的命令，那么这个命令就非常有用；如果用户是超级用户，那么还可以使用 nice 来提升某个命令的优先级
nohup	允许用户在不结束后台运行进程的情况下注销；在某些系统中，这一项是自动设置的
notify	让 Shell 在某个作业的状态发生改变时，立即通知用户
onintr	控制脚本遇到中断时采取的动作
popd	从目录栈中删除一个目录
printenv	显示所有环境变量的名称和取值
pushd	改变工作目录，并将新的目录放置到目录栈的顶部
rehash	重新创建 hash 机制使用的内部表格；每当调用新的 tcsh 实例时，hash 机制都会创建一个基于 path 值的所有可用命令的有序列表；当用户向 path 中的某个目录添加一个新的命令时，使用 rehash 重新创建这个命令有序表；如果没有这样做，那么 tcsh 可能将搜索不到这个新的命令；还可以参见 hashstat 和 unhash

续表

内置命令	功　　能
repeat	带有两个参数，即次数 count 和简单命令（即不包含管道和命令列表的命令），将该命令重复执行 count 次
sched	在某个指定的时间执行命令
set	声明、初始化和显示局部变量
setenv	声明、初始化和显示环境变量
shift	与 bash 的 shift 内置命令类似；如果不带任何参数，shift 将增加 argv 数组的索引。还可以带上一个数组名作为参数，对该数组执行同样的操作
source	执行参数所指定的 Shell 脚本：source 并不另外创建一个新的进程，这个与 bash 的句号（.）内置命令类似；因为 source 内置命令执行的是 TC Shell 脚本，所以在脚本起始处不需要以#!开头；因为是当前 Shell 执行 source，所以包含诸如 set 之类的命令的脚本将影响当前 Shell；在用户修改完.tcshrc 或者.login 文件之后，可以在 Shell 中使用 source 来执行它们，这样就可以在不注销并再次登录的情况下使修改生效；可以嵌套使用 source 内置命令
stop	停止后台运行的某个作业或者进程，stop 内置命令可以接受多个参数
suspend	停止当前 Shell 并将其放置到后台，与停止前台运行作业的挂起键类似
time	执行参数指定的命令；根据 Shell 变量 time，这条命令将显示关于这条命令的时间信息；如果不带任何参数，那么 time 将显示当前 Shell 与其子 Shell 的时间
umask	指定或者修改用户所创建的文件的访问权限
unalias	删除别名
unhash	关闭 hash 机制，还可参见 hashstat 和 rehash
unlimit	删除对当前进程的限制
unset	删除变量声明
unsetenv	删除环境变量声明
wait	让 Shell 等待所有子进程结束；如果在提示符后面输入 wait 命令，那么直到所有后台进程全部执行完毕之后，tcsh 才显示提示符；如果使用中断键中断 wait，那么它将在 tcsh 显示提示符之前显示正在运行的进程列表
where	如果使用某条命令的名字作为参数，那么 where 将定位该命令的所有出现，并且针对每次出现，都将指出它是别名、内置命令还是位于用户的 path 变量中的某个可执行文件
which	与 where 类似，但是只报告将要执行的那条命令，而不是所有可能出现的命令；这个内置命令的速度要比 Linux 工具 which 快得多，并且能够识别别名和内置命令

电子工业出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY http://www.phei.com.cn Broadview WWW.BROADVIEW.COM.CN

博文视点 · IT出版旗舰品牌

博文视点诚邀精锐作者加盟

十载耕耘奠定专业地位

以书为证彰显卓越品质

《C++Primer（中文版）（第5版）》、《淘宝技术这十年》、《代码大全》、《Windows内核情景分析》、《加密与解密》、《编程之美》、《VC++深入详解》、《SEO实战密码》、《PPT演义》……

“圣经”级图书光耀夺目，被无数读者朋友奉为案头手册传世经典。

潘爱民、毛德操、张亚勤、张宏江、昝辉Zac、李刚、曹江华……

“明星”级作者济济一堂，他们的名字熠熠生辉，与IT业的蓬勃发展紧密相连。

十年的开拓、探索和励精图治，成就**博**古通今、**文**圆质方、**视**角独特、**点**石成金之计算机图书的风向标杆：博文视点。

“凤翱翔于千仞兮，非梧不栖”，博文视点欢迎更多才华横溢、锐意创新的作者朋友加盟，与大师并列于IT专业出版之巅。

英雄帖

江湖风云起，代有才人出。
IT界群雄并起，逐鹿中原。
博文视点诚邀天下技术英豪加入，
指点江山，激扬文字
传播信息技术，分享IT心得

• 专业的作者服务 •

博文视点自成立以来一直专注于 IT 专业技术图书的出版，拥有丰富的与技术图书作者合作的经验，并参照IT技术图书的特点，打造了一支高效运转、富有服务意识的编辑出版团队。我们始终坚持：

善待作者——我们会把出版流程整理得清晰简明，为作者提供优厚的稿酬服务，解除作者的顾虑，安心写作，展现出最好的作品。

尊重作者——我们尊重每一位作者的技术实力和生活习惯，并会参照作者实际的工作、生活节奏，量身制定写作计划，确保合作顺利进行。

提升作者——我们打造精品图书，更要打造知名作者。博文视点致力于通过图书提升作者的个人品牌和技术影响力，为作者的事业开拓带来更多的机会。

联系我们

博文视点官网：http://www.broadview.com.cn　CSDN官方博客：http://blog.csdn.net/broadview2006/

投稿电话：010-51260888　88254368　投稿邮箱：jsj@phei.com.cn

新浪微博 weibo.com @博文视点Broadview

微信公众账号 博文视点Broadview

電子工業出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY http://www.phei.com.cn

Broadview® WWW.BROADVIEW.COM.CN

博文视点·IT出版旗舰品牌

博文视点精品图书展台

专业典藏

移动开发

大数据·云计算·物联网

数据库

Web开发

程序设计

软件工程

办公精品

网络营销

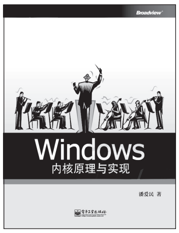
電子工業出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY http://www.phei.com.cn
Broadview® WWW.BROADVIEW.COM.CN

博文视点·IT出版旗舰品牌

博文视点本季最新最热图书

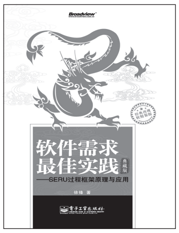
《淘宝技术这十年》

子柳 著
定价：45.00元
◎ 淘宝技术大学校长辛辣揭秘
◎ 世界最大电商平台、超大型网站，首次全面曝光技术内幕
◎ 技术变迁熠熠生辉、产品演进饱含智慧、牛人生涯叱咤风云、圈内趣事令人捧腹



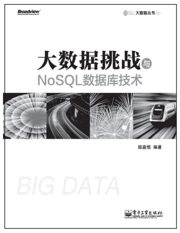
《Windows内核原理与实现》

潘爱民 著
定价：99.00 元
第一本用真实的源代码剖析 Windows 操作系统核心原理的原创著作！



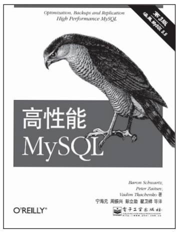
《软件需求最佳实践——SERU过程框架原理与应用（典藏版）》

徐锋 著
定价: 69.00元
"用户说不清需求"、"需求变更频繁"……，都是在软件需求实践中频繁遇到的问题；本书首先直面这些问题，从心理学、社会学的角度剖析其背后的深层原因，使大家从中获得突破的方法。



大数据丛书
《大数据挑战与NoSQL数据库技术》

陆嘉恒 编著
定价：79.00元
大数据技术的学习指南。突破迷局，厘清思路，拥抱变化。

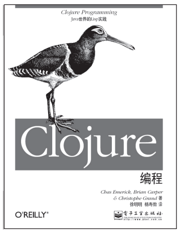


《高性能MySQL（第3版）》

【美】施瓦茨（Schwartz,B.）【美】扎伊采夫（Zaitsev,P.）【美】特卡琴科（Tkachenko,V.）著
宁海元 周振兴 彭立勋等 译
定价：128.00元
本书是 MySQL 领域的经典之作，拥有广泛的影响力。第 3 版更新了大量的内容，不但涵盖了最新 MySQL 5.5 版本的新特性，也讲述了关于固态盘、高可扩展性设计和云计算环境下的数据库相关的新内容，原有的基准测试和性能优化部分也做了大量的扩展和补充。

《收获，不止Oracle》

梁敬彬 梁敬弘 著
定价：59.00元
颠覆 IT 技术图书的传统写作方式，在妙趣横生的故事中学到 Oracle 核心知识与优化方法论，让你摆脱技术束缚，超越技术。



《Clojure编程》

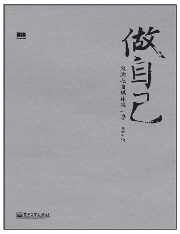
[美]Chas Emerick（蔡司 埃默里克）,Brian Carper(布赖恩 卡珀),Christophe Grand（克里斯托弗 格兰德）著
徐明明 杨寿勋 译
定价：99.00元
第一本完整讲述 Cloujore 的权威著作。

百度认证系列丛书
《百度推广——搜索营销新视角》

百度营销研究院 著
定价：59.00元
百度营销研究院资深专家团队撰写，百度认证初级教程！

《我看电商》

黄若 著
定价：39.00元
作者近三十年从事零售及电子商务管理的总结和分享。



《做自己——鬼脚七自媒体第一季》

鬼脚七 著
定价：77.00元
本书是鬼脚七自媒体的原创文集，是电商圈第 1 本自媒体著作！书中有关于生活、互联网、自媒体的睿智分享，也有关于淘宝、搜索的独到见解。文章非常耐读，也容易引发读者的思考，是一本接地气、文艺范、充满正能量的电商生活书！

欢迎投稿：

投稿信箱：jsj@phei.com.cn
editor@broadview.com.cn
读者信箱：market@broadview.com.cn
电　　话：010-51260888

更多信息请关注：
博文视点官方网站：
http://www.broadview.com.cn
博文视点官方微博：
http://t.sina.com.cn/broadviewbj